ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

Σχεδίαση διαδραστικών dashboards με δεδομένα από API

Ηλίας-Ιάσων Κατσούτας-Καλομοίρης

Αθήνα, 2022

HAROKOPIO UNIVERSITY

ATHENS HAROKOPIO UNIVERSITY
INFORMATICS AND TELEMATICS DEPARTMENT

Interactive dashboard design with data from API

Ilias-Iason Katsoutas-Kalomoiris

Athens, 2022

ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

Τριμελής Εξεταστική Επιτροπή

Ηρακλής Βαρλάμης (Επιβλέπων)
Αναπληρωτής Καθηγητής
Τμήματος Πληροφορικής και τηλεματικής,
Χαροκόπειο πανεπιστήμιο

Χρήστος Δίου
Επίκουρος Καθηγητής
Τμήματος Πληροφορικής και τηλεματικής,
Χαροκόπειο πανεπιστήμιο

Δημήτριος Μιχαήλ
Αναπληρωτής Καθηγητής
Τμήματος Πληροφορικής και τηλεματικής,
Χαροκόπειο πανεπιστήμιο

Ο Ηλίας-Ιάσων Κατσούτας-Καλομοίρης,

δηλώνω υπεύθυνα ότι:

1. Είμαι ο κάτοχος των πνευματικών δικαιωμάτων της πρωτότυπης αυτής εργασίας και από όσο γνωρίζω η εργασία μου δε συκοφαντεί πρόσωπα, ούτε προσβάλει τα πνευματικά δικαιώματα τρίτων.

2. Αποδέχομαι ότι η ΒΚΠ μπορεί, χωρίς να αλλάξει το περιεχόμενο της εργασίας μου, να τη διαθέσει σε ηλεκτρονική μορφή μέσα από τη ψηφιακή Βιβλιοθήκη της, να την αντιγράψει σε οποιοδήποτε μέσο ή/και σε οποιοδήποτε μορφότυπο καθώς και να κρατά περισσότερα από ένα αντίγραφα για λόγους συντήρησης και ασφάλειας.

Αφιέρωση

Στην όλη οικογένεια μου

Ευχαριστίες

Με την ολοκλήρωση της πτυχιακής μου, θα ήθελα να ευχαριστήσω όλους όσους με βοήθησαν και με στήριξαν κατά την διάρκεια των σπουδών μου, αλλά και κατά την εκπόνηση της εργασίας αυτής.

Η ολοκλήρωση της πτυχιακής αυτής εργασίας δεν θα ήταν δυνατή χωρίς την υποστήριξη του καθηγητή μου, κ Βαρλάμη, που με καθοδήγησε και με βοήθησε κατα την δημιουργία αυτης της εργασίας. Επιπλέον, θα ήθελα να ευχαριστήσω όλους τους καθηγητές μου και ολόκληρο το προσωπικό του Χαροκόπειου Πανεπιστημίου για την βοήθεια και τις γνώσεις που μου προσέφεραν καθ' όλη την διάρκεια της φοίτησης μου.

Τέλος, θέλω να ευχαριστήσω πολύ τους γονείς μου , οι οποίοι υπήρξαν στήριγμα για μένα και στους οποίους οφείλω ολη την διαδρομή των σπουδών μου και όχι μόνο.

Περιεχόμενα

Περίληψη

Τα γραφήματα και τα διαγράμματα είναι τα πιο σημαντικά εργαλεία, οταν πρόκειται για ανάγνωση πολύπλοκων και συμπυκνωμένων δεδομένων. Αποτελούν απλοποιημένες και δομημένες οπτικες παρουσιάσεις σχέσεων, στατιστικών δεδομένων κλπ. Με την χρήση τους, δίνεται η δυνατότητα στον χρήστη να κατανοήσει και να απομνημονευσει πιο ευκολα τα δεδομένα, καθώς και να παρατηρήσει με μεγαλύτερη ευκολία μοτίβα και συσχετισμούς. Υπάρχουν πολλαπλά είδη διαγραμμάτων, καθένα από τα οποία στοχεύει σε διαφορετικό σκοπό.

Το αντικείμενο της πτυχιακής αυτής εργασίας, είναι η σχεδίαση δυναμικών/διαδραστικών διαγραμμάτων και dashboards, με πολυπληθή πολύπλοκα δεδομένα, τα οποία προέρχονται από ανοιχτό API κρυπτονομισμάτων. Στόχος της είναι ο χρήστης να έχει την δυνατότητα να μελετήσει δεδομένα που τον ενδιαφέρουν και να παραμετροποιήσει τις απεικονίσεις αναλογα με τις προτιμήσεις του.

Η αναπαράσταση δεδομένων με γραφήματα έγινε δυνατή με την χρήση της γραμματικής Vega-lite, η οποία αποτελεί μια απλουστευμένη εκδοχή του Vega,. Η Vega είναι μια γλώσσα που παρέχει τα βασικά δομικά στοιχεία για την δημιουργία σύνθετων διαγραμμάτων/γραφημάτων, οι προδιαγραφές των οποίων οριζονται με μορφή JSON.

Περίληψη στα Αγγλικά

Graphs and diagrams are one of the most important tools, when it comes to reading and understanding complex and concentrated data. Those entities are simplified and structured visual presentations of relations, statistical data etc. Their usage gives the user the opportunity to understand, apprehend and memorize data effortlessly, since it's easier to spot patterns and relations. There are numerous types of diagrams, and each one of them serves a different purpose.

The subject of this thesis statement is the creation of dynamic/interactive dashboards and diagrams, with complex and concentrated data that originate from a public API of cryptocurrencies. The goal of this thesis is to give the user the capability to study data that interest him and customize the visualizations depending on his needs.

The data visualization is possible with the usage of the VEGA LITE grammar, which is a simplification of the original VEGA. VEGA is a language that provides the basic structure for the creation of complex diagrams and graphs, the specifications of which are written in JSON format.

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΊΕΣ

HTTP	Text Transfer Protocol
CSS	Cascading Style Sheets
API	Application Programming Interface
REST	Representational state transfer
SCSS	Sassy Cascading Style Sheets
JSS	JavaScript Style Sheets
ES6	ECMAScript 6
JSX	JavaScript XML
DOM	Document Object Model
SQL	Structured query language

ΚΕΦΑΛΑΙΟ 1: Εισαγωγη

1.1 Σκοπός εργασίας

Σκοπός της πτυχιακής εργασίας είναι η υλοποίηση διαδικτυακής εφαρμογής διαδραστικών διαγραμμάτων και dashboard, η οποία τροφοδοτείται με δεδομένα από ανοιχτό API κρυπτονομισμάτων. Με την βοήθεια τεχνολογιών οπτικοποίησης και διαφόρων εργαλείων - βιβλιοθηκών για την ανάπτυξη ιστοσελίδων, έχω ως στόχο να δημιουργήσω ένα περιβάλλον όπου ο χρήστης μπορεί να παρατηρήσει, να συσχετίσει και να συγκρίνει δεδομένα μεταξύ κρυπτονομισμάτων, οδηγώντας τον στην καλύτερη κατανόηση των δεδομένων αυτών.

- API url: https://github.com/jasonKatsman/it21634-thesis-api
- Front end application url: https://github.com/jasonKatsman/it21634-thesis-app

1.2 Δομή εργασίας

Η παρούσα πτυχιακή εργασία είναι διαρθρωμένη ως εξής:

- Στο Κεφάλαιο 1 γίνεται ο προσδιορισμός του σκοπού και της δομής της εργασίας.
- Στο κεφάλαιο 2 γίνεται επεξήγηση των εργαλείων, των γλωσσών, των τεχνολογιών και των βιβλιοθηκών που χρησιμοποιήθηκαν για τις εφαρμογές
- Στο κεφαλαιο 3 γίνεται ανάλυση για τους τύπους ταμπλό και διαγραμμάτων που χρησιμοποιήθηκαν
- Στο κεφαλαιο 4 αναλύεται η σχεδίαση και η αρχιτεκτονική των εφαρμογών
- Στο κεφάλαιο 5 εξηγείται η υλοποίηση των εφαρμογών, η δομή των αρχείων, και εξηγούνται κομμάτια κώδικα των εφαρμογών.
- Στο κεφάλαιο 6 πραγματοποιείται η ανάλυση των σεναρίων χρήσης της εφαρμογής.
- Στο κεφάλαιο 7, γίνεται μια περιεκτική παρουσίαση των συμπερασμάτων

ΚΕΦΑΛΑΙΟ 2: Εργαλεία και τεχνολογίες

2.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα γίνει μια συνοπτική περιγραφή των τεχνολογιών και βιβλιοθηκών που χρησιμοποιήθηκαν για την υλοποίηση αυτής της πτυχιακής εργασίας. Στην εφαρμογή που υλοποιήθηκε, χρησιμοποιήθηκε javascript και typescript ως γλώσσα προγραμματισμού. Για την δημιουργία του back end έγινε χρήση του nodeJS και η αξιοποίηση της βιβλιοθήκης express. Για το front end, χρησιμοποιήθηκε η βιβλιοθήκη reactJS,συνοδευτικά με την χρήση HTML5 και JSS (εργαλείο συγγραφής css με χρήση javascript), και η αξιοποίηση ενός UI framework, του material UI. Η βάση δεδομένων υλοποιήθηκε σε noSQL, και συγκεκριμένα στην βαση Mongodb.

2.2 HTML

Το html αποτελεί μια γλώσσα σήμανσης, η οποία είναι σχεδιασμένη να τρέχει σε προγράμματα περιήγησης(web browsers) και η ονομασία αποτελεί συντομογραφία του Hypertext Markup Language (Γλώσσα Σήμανσης Υπερκειμένου). Συνήθως, το html συνοδεύεται με τεχνολογίες όπως το CSS(Cascading Style Sheets) και την γλώσσα προγραμματισμού javascript.

Η html χρησιμοποιεί στοιχεία(elements) για να περιγράψει την δομή μιας σελίδας. Κάθε element αποτελείται συνήθως από δύο ετικέτες (tags), την ετικέτα εναρξης (opening tag) και την ετικέτα λήξης (closing tag). Οι χαρακτήρες μέσα στις αγκύλες μιας ετικέτας υποδεικνύουν τον σκοπό της ετικέτας αυτής. Για παράδειγμα, στην ετικέτα <p>, το p σηματοδοτεί οτι αυτη η ετικέτα είναι παράγραφος (paragraph). Αντίστοιχα, το <h1> ως τιτλος (header) κτλ. Η ετικέτα λήξης ξεχωρίζει, έχοντας μία κάθετο πρίν τον χαρακτήρα. Επιπλέον, κάθε στοιχείο διαθέτει χαρακτηριστικά (attributes), τα οποία παρέχουν επιπρόσθετες πληροφορίες το στοιχείο αυτό. Τα χαρακτηριστικά αυτά περιλαμβάνονται στην ετικέτα ανοίγματος (opening tag).

Βασικά δομικά στοιχεία σε μία σελίδα HTML είναι το head και το body. Το body, είναι το πιο βασικό στοιχείο σε ενα HTML αρχείο, καθώς τα περιεχόμενα του είναι αυτά που εμφανίζονται στο κύριο παράθυρο του προγράμματος περιήγησης. Το head Συνήθως βρίσκεται πάνω από το body, και περιέχει διάφορες πληροφορίες για την σελίδα. Μέσα στο head, υπάρχει συνήθως και το στοιχείο title, το περιεχόμενο του οποίου
εμφανίζεται στην καρτέλα του προγράμματος περιήγησης.

Η HTML βρίσκεται στην 5η της έκδοση (HTML5). Με την έκδοση 5, έχει προστεθεί μεγάλο πλήθος στοιχείων, καθώς και έχουν τροποποιηθεί ή αφερεθεί μερικά παλιά στοιχεία. Η νέα έκδοση διαθέτει στοιχεία για βίντεο και ήχο, τα οποία μπορούν να ενταχθούν απευθείας σε μια HTML σελίδα, σε σύγκριση με πριν, που απαιτούνταν χρήση adobe flash player. Είναι φιλική στις κινητές συσκευές, έχει ευρεία υποστήριξη από περιηγητές κτλ.

```
<html lang="en-US">
    <head>
        <title>This is a title</title>
    </head>
    <body>
        <h1>Hello World!</h1>
        <p>This is a paragraph.</p>
    </body>
</html>
```

Εικόνα 1: Παράδειγμα HTML

2.3 CSS

Το css (cascading style sheets) είναι γλώσσα που καθορίζει την εμφάνιση των στοιχείων σε μία σελίδα html. Με την βοήθεια της μπορεί να καθοριστεί το χρώμα, η διάταξη, το σχήμα, το μέγεθος και άλλες ιδιότητες των στοιχείων της σελίδας.

Το css αποτελείται από κανόνες (rules). Κάθε κανόνας σχετίζεται με στοιχεία της σελίδας, και προσδιορίζει πώς θα εμφανίζονται τα συγκεκριμένα αυτα στοιχεία. Ένας κανόνας αποτελείται από δύο μέρη, τον selector και το declaration. Οι selectors υποδεικνύουν σε ποιο στοιχείο θα εφαρμοστεί ο κανόνας. Ο ίδιος κανόνας μπορεί να εφαρμόζεται σε πολλαπλά στοιχεία, εφόσον χωριστούν με κόμμα (,). Υπάρχουν πολλαπλά είδη selectors, τα οποία στοχεύουν σε διαφορετικά στοιχεία. Για παράδειγμα, μπορούμε να επιλέξουμε στοιχεία με βάση το όνομα τους, με βάση το class ή το id attribute τους, κτλ.Τα declarations υποδεικνύουν πως τα στοιχεία που αναφέρονται με τον selector θα πρέπει να σχεδιαστούν. Ενα declaration αποτελείται από δύο μέρη. Το πρώτο μέρος λέγεται property, το οποίο καθορίζει την ιδιότητα, και το δευτερο μερος λέγεται value, το οποίο καθορίζει τις ρυθμίσεις. Για παράδειγμα, το property μπορεί να είναι το color, και το value να είναι dark green. (color: darkgreen)

```css
h1, h2, p {
    font-family: sans-serif;
    color: darkgreen;
}
```

Εικόνα 2: Παράδειγμα CSS

Η γραφή του css γίνεται με δύο τρόπους, είτε σε αρχείο ξεχωριστό απο του html, είτε μέσα στο html αρχείο. Για το πρώτο, απαιτείται ή δημιουργία αρχείου .css, και για να επιδράσει σε κάποιο html αρχείο, πρέπει να συμπεριληφθεί μέσω του στοιχείου <link>, προσδιορίζοντας την τοποθεσία του αρχείου. Αλλιώς, η συγγραφή του css εντός ενός αρχείου html γίνεται μέσα στο στοιχείο <style>, το οποίο μπαίνει κάτω από το body.[12]

Η πίο προηγμένη μορφή του CSS είναι το SCSS (Sassy Cascading Style Sheets) και αποτελεί γλώσσα που μεταγλωτίζεται σε CSS. Το SCSS παρέχει πολλές επιπλέον λειτουργίες σε συγκριση με το απλό CSS, όπως μεταβλητές (variables), εμφωλευμένα selections (nesting). Αυτό καθιστά την SCSS πολύ πιο απλή και γρήγορη στη γραφή σε σύγκριση με την CSS.

```scss
.box {
    background: darkgreen;
    &:hover {
        background:gray
    }

}
```

Εικόνα 3: Παράδειγμα SCSS

Τέλος, η JSS, αποτελεί εργαλείο που επιτρέπει την χρήση javascript γία να περιγράψει στυλ, με τρόπο δηλωτικό. Για την χρήση του απαιτείται η εγκατάσταση πακέτων.

```jss
const styles = {
    color: `${color}`,
    background: 'green',
    '&:hover' {
        background:'gray'
    }
}
```

Εικόνα 4: Παράδειγμα JSS

[1] https://en.wikipedia.org/wiki/CSS
[2] HTML & CSS Design and Build Websites Jon Duckett John Wiley & Sons, Inc.

2.4 JAVASCRIPT

Η javascript είναι γλώσσα προγραμματισμού που χρησιμοποιείται ευρέως για την δημιουργία διαδικτυακών εφαρμογών και κατ επέκταση για δημιουργία πάσης φύσεως προγραμμάτων. Αποτελεί αντικειμενοστραφής γλώσσα προγραμματισμού σχεδιασμένη το 1995 απο τον Brendan Eich, για να επιτρέπει στους μη προγραμματιστές να επεκτείνουν την λειτουργικότητα των ιστοσελίδων, με client side εκτελέσιμο κώδικα. Αντίθετα με τις παραδοσιακές γλώσσες προγραμματισμού, οπως την C# και την Java, δεν διαθέτει κλάσεις, και δεν απαιτεί δομημένο προγραμματισμό, κανοντάς την πολύ ευελικτη γλώσσα.[3]

Η πιο πρόσφατη έκδοση ECMAscript είναι η 6η (ES6). Κάθε έκδοση προσφέρει νέα χαρακτηριστικά και αλλαγές. Η 6η εκδοση της ECMAscript φέρνει πολλά νέα χαρακτηριστικά σε συγκριση με παλαιότερες εκδόσεις, οπως arrow functions, κλάσεις, template strings, enhanced Object Literals, destructuring, ,αντικατάσταση του var με του let, promises, symbols, generators και άλλα.

```
const addNumbers = (first, second) => {
    return first + second;
}
```

Εικόνα 5: Παράδειγμα Javascript ES6 με arrow function

```
<script>
    document.querySelector("#head").addEventListener("click", removeRow)

    function removeRow() {
        const row = document.querySelector("#row");
        row.parentNode.removeChild(row);
    }
</script>
```

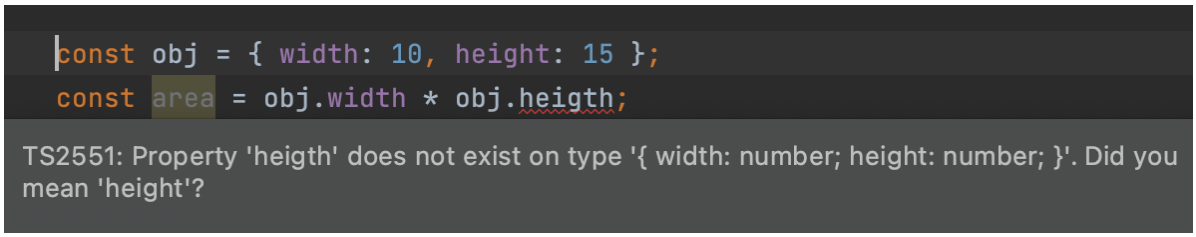
Εικόνα 6: Παράδειγμα Javascript σε HTML αρχείο

[3] An Analysis of the Dynamic Behavior of JavaScript Programs Gregor Richards Sylvain Lebresne Brian Burg Jan Vitek S3 Lab, Department of Computer Science, Purdue University, West Lafayette, IN {gkrichar,slebresn,bburg,jv}@cs.purdue.edu

2.5 TYPESCRIPT

Η TypeScript αποτελεί strongly typed γλώσσα προγραμματισμού (απαιτεί αναθέσεις τύπου στις μεταβλητές) που έχει χτιστεί πάνω στην JavaScript. Η TypeScript προσθέτει επιπλέον σύνταξη στην JavaScript, για να υποστηρίξει πιο αυστηρό integration με τον editor, έχοντας ώς αποτέλεσμα την αποφυγή σφαλμάτων. Ο κώδικας της γλώσσας TypeScript μετατρέπεται σε γλώσσα Javascript, η οποία τρέχει οπουδήποτε μπορεί να τρέξει ήδη η JavaScript, οπως στον browser, σε Node.js κτλ.

Η TypeScript ελέγχει ένα πρόγραμμα για σφάλματα πριν απο την εκτέλεση του, με βάση τον τύπο των τιμών/ μεταβλητών (static type checker).



Εικόνα 7. Παράδειγμα Typescript type checking

Η Γλώσσα αυτή είναι υπερσύνολο της Javascript, επομένως η σύνταξη της javaScript είναι επιτρεπτή. Ετσι, ο κώδικας της JavaScript μπορεί απευθείας να τρέξει. Παρόλα αυτα, η TypeScript είναι ένα typed υπερσύνολο, οπου αυτό σημαίνει ότι προσθέτει παραπάνω κανόνες σχετικά με το πως διάφοροι τύποι τιμών μπορούν να χρησιμοποιηθούν. Για παράδειγμα, η typeScript δεν θα μάς αφήσει να διαιρέσουμε έναν αριθμό με ένα string.

Με την TypeScript, γίνεται δυνατόν να δημιουργήσουμε τύπους (types). Σε περίπτωση που δέν υπάρχει τύπος σε μία μεταβλητή, τότε παίρνει τον τύπο της τιμης. Επιπλέον, είναι δυνατή η σύνθεση πολλαπλών τύπων και η δημιουργία interfaces. Για την χρήση της TypeScript απαιτείται εγκατάσταση πακέτου.[4]

[4] https://www.typescriptlang.org/

```
type Customer = {
    fullName: string;
    age: number;
    phone?: number;
}

const customerA: Customer = {
 fullName: "Jason K.",
 age: 25
}
```

Εικόνα 8. Παράδειγμα Typescript ανάθεση τύπου

2.6 NODEJS

Το NodeJS, είναι ένα server side JavaScript περιβάλλον. Είναι βασισμένη στην V8 (V8 JavaScript Engine) της Google. Η node είναι υλοποιημένη κυρίως σε C και C++, εστιάζοντας στην απόδοση και στην χαμηλή χρήση μνήμης. Αντίθετα με τα υπόλοιπα μοντέρνα περιβάλλοντα, η nodeJS δεν είναι πολυνηματική. Είναι βασισμένο σε ένα μοντέλο ασύγχρονης επικοινωνίας εισόδου και εξόδου. Η javascript είναι η ιδανικη επιλογή για αυτή την λογική, καθώς η ίδια υποστηρίζει callbacks. Η λογική της javaScript ευνοεί την δημιουργία callback συναρτήσεων που μπορούν να καταχωρηθούν σαν event handlers.

Το Node είναι από τα πιό γνωστά frameworks και περιβάλλοντα που υποστηρίζουν ανάπτυξη server side javascript εφαρμογών. Έχει μεγάλο κοινό, το οποίο συμβάλλει στην δημιουργία βιβλιοθηκών που μπορούν να λειτουργήσουν σε αυτό. Υπάρχουν αμέτρητα εργαλεία και βιβλιοθήκες και web stack που διευκολύνουν στην δημιουργία web εφαρμογών και όχι μόνο. Επιπλέον, υπάρχει διαθεσιμότητα βιβλιοθηκών και για την client side javaScript. Το node package manager (npm) αποτελεί το εργαλείο που δίνει την δυνατότητα εγκατάστασης πακέτων.

Κάθε ενέργεια εισόδου/εξόδου αντιμετωπίζεται από higher - order συναρτήσεις, δηλαδή συναρτήσεις που πέρνουν συναρτήσεις σαν παραμέτρους, που καθορίζουν τί πρέπει να συμβεί μια καθορισμένη στιγμή. Σε μία ασύγχρονη διεργασία (πχ που απαιτεί σύνδεση με βάση

δεδομένων, δικτυο κτλ), όταν τα δεδομένα γίνουν διαθέσιμα, ή αν συμβεί ένα σφάλμα, τότε θα πυροδοτηθεί ή ανάλογη callback συνάρτηση.[5]

Το express αποτελεί framework το οποίο διαθέτει προκατασκευασμένα εργαλεία που βοηθούν στην δημιουργία server side δικτυακών εφαρμογών. Παρέχει μηχανισμούς με τους οποίους μας δίνεται η δυνατότητα να γράψουμε handlers για requests που έχουν διαφορα paths και HTTP μεθόδους (POST PUT PATCH DELETE GET). Επιπλέον, υπάρχει η δυνατότητα να προσδιορίσεις το port για την πραγματοποίηση της σύνδεσης, και την τοποθεσία των συναρτήσεων που χρησιμοποιούνται για την δημιουργία των responses. Εκτός από αυτό, γίνεται να προσθέσουμε middleware σε οποιοδήποτε σημείο ενός request.

```
app.get('/', async function(request, response) {
    try {
        const res = await mongodb.getCandleStickData()
        return response.json(res)
    } catch (e) {
        return response.status(500).send(e)
    }
});
```

Εικόνα 9.Παράδειγμα Express *route definition*

2.7 REACTJS

Η ReactJs αποτελεί μια JavaScript βιβλιοθήκη, που σκοπός της είναι η δημιουργία user interfaces. Η βιβλιοθήκη αυτή είναι ανοιχτού κώδικα, φτιαγμένη απο τον Jordan Walke, software engineer της Facebook. Είναι εύκολο με αυτήν να φτιαχτούν δυναμικές εφαρμογές, καθώς απαιτεί λιγότερο κώδικα και προσφέρει μεγάλη λειτουργικότητα, σε αντίθεση με την απλή javascript, που γίνεται πολύπλοκη όταν πρόκειται για σύνθετες εφαρμογές.

Το JSX (JavaScript Syntax Extension) είναι επέκταση σύνταξης της JavaScript. Χρησιμοποιείται απο την React για να περιγράψει πώς θα δείχνει ενα UI. Με την χρήση JSX, γίνεται δυνατή η εγγραφή HTML στο ίδιο αρχείο που περιέχει κώδικα JavaScript. Αυτό κάνει

[5] Tilkov, S.; Vinoski, S. (2010). Node.js: Using JavaScript to Build High-Performance Network Programs. , 14(6), 80–83. doi:10.1109/mic.2010.145

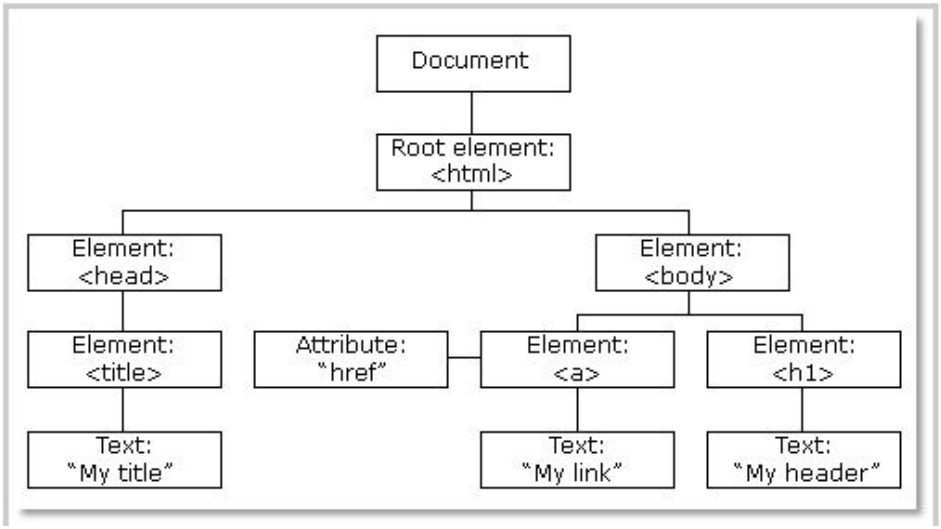
τον κώδικα πιο κατανοητό και εύκολο στην αποσφαλμάτωση. Ουσιαστικά ενσωματώνει στην HTML Javascript κώδικα.

```
const values = [...];

const ListItems = values.map((item :{name: string} , i :number ) => (
  <div key={i}>
    <p className={'nameText'}>{item.name}</p>
  </div>
));
```

Εικόνα 10. Παράδειγμα JSX σε React

Το DOM (Document Object Model) είναι μια αναπαράσταση της σελίδας, που κατασκευάζεται οταν φορτώνει η σελίδα και αποτελείται από DOM nodes, καθένα από τα οποία αποτελεί κάποιο HTML element. Κάθε node μπορεί να είναι πατέρας κάποιων node παιδίων, αμα υπαρχουν κάτω από αυτό και μπορεί να έχει αδέρφια, αν βρίσκονται στο ίδιο επίπεδο.



Εικόνα 11. Παράδειγμα DOM

Η React διατηρεί μία αναπαράσταση του πραγματικού DOM στην μνήμη, που είναι γνωστή ως Virtual DOM. Η αλλαγή του πραγματικού DOM είναι πολύ πιο αργή από την αλλαγή του VDOM. ‘Οταν η κατάσταση ενός object αλλάζει, τοτε το VDOM αλλάζει μόνο το συγκεκριμένο στοιχείο στο πραγματικό DOM, χωρίς να ανανεώσει τα υπόλοιπα στοιχεία. Η

ανανέωση αυτή στην ReactJS γίνεται δυνατή μέσω της βιβλιοθήκης ReactDOM, και η διαδικασία αυτή είναι γνωστή ως reconciliation.

Η React διαθέτει πολλά εργαλεία, τα οποία επεκτείνουν την λειτουργικότητα της. Συγκεκριμένα, προσφέρει server side rendering, μέσω της NextJS, παρέχει state management εργαλεία (πχ Redux). Τελος, υπάρχει και το Framework React Native, το οποίο χρησιμοποιείται για την δημιουργία εφαρμογών για κινητές συσκευές. [6]

Η ReactJS χωρίζει το UI σε διάφορα components, τα οποία είναι επαναχρησιμοποιήσιμα. Αυτά τα components είναι σαν javascript συναρτήσεις, τα οποία δέχονται παραμέτρους που ονομάζονται props, και επιστρέφουν στοιχεία που περιγράφουν τι πρέπει να εμφανίζεται στην οθόνη. Τα components χωρίζονται σε class ή functional, τα οποία έχουν την ίδια λειτουργικότητα, απλά με διαφορετική δομή. Αυτά τα components μπορούν να χρησιμοποιηθούν εμφωλευμένα σε αλλα components. Για να φορτωθεί στο DOM ενα component, θα πρέπει το στοιχείο (ή ο πατέρας αυτου) να υπάρχει σαν παράμετρος στο ReactDOM.render()

```
const FunctionComponent = (props) => {
  return <p>{props.text}</p>;
};
```

Εικόνα 12. Παράδειγμα functional Component

```
class FunctionComponent extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

Εικόνα 13. Παράδειγμα class Component

Η React διαθέτει hooks, τα οποία βοηθούν στην διαχείριση state και άλλων χαρακτηριστικών των functional component. Είναι συναρτήσεις που σου επιτρέπουν να

[6] https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs#what_is_react

αποθηκεύσεις state και αλλα lifeCycle χαρακτηριστικά. Τα πιό συνηθισμένα hooks είναι το state και το effect.

- Το useState hook καλείται μέσα σε καποιο component για την διαχείριση του local state του. Δέχεται σαν argument την αρχική τιμή του. Επιστρέφει 2 τιμές, την τρέχουσα τιμή του state και μία συνάρτηση που την ανανεώνει. Η αλλαγή αυτου του state προκαλεί re-render του συγκεκριμένου component, χωρίς όμως αυτό να χάσει τις τιμές του.

- Το useEffect αποτελεί μια συνάρτηση η οποία τρέχει μετά το render του component, πυροδοτόντας μια ενεργεια. Διαθέτει array παράμετρο που μπορεί να παραλείψει την πυροδότηση του, σε περίπτωση που κάποια από τις συνθήκες που συμπεριλαμβάνονται στο array είναι ψευδείς ή αν η τιμή δεν έχει αλλάξει σε σχέση με το προηγούμενο re-render.[7]

```
const CustomInput = () => {
  const [name, setName] = useState( initialState: '');

  useEffect( effect: () => {
    console.log('i run if the name is changed!');
  }, deps: [name]);

  const onChange = (e) => {
    setName(e.target.value);
  };

  return <input value={name} onChange={onChange} />;
};
```

Εικόνα 14. Παράδειγμα Component με hooks

Η react διαθέτει πολλά εργαλεία και επεκτάσεις που μπορούν να χρησιμοποιηθούν για την ευκολότερη δημιουργία εφαρμογών και λειτουργιών. Συγκεκριμένα, διαθέτει εργαλεία που βοηθούν στο state management όπως τη Redux, και διαθέτει frameworks που είναι βασισμένα σε αυτήν, οπως την NextJs και react Native.

[7] https://reactjs.org/

2.8 MATERIAL UI

Το MATERIAL UI είναι βιβλιοθήκη που παρέχει έτοιμα styled components, με δυνατότητα τροποποίησης, που σκοπεύει στην ταχύτερη δημιουργία react εφαρμογών καθώς δεν χρειάζεται η δημιουργία κάποιων component από την αρχή. Το material UI είναι βασισμένο στο material design της google.

Αυτή η βιβλιοθήκη διαθέτει εργαλεία που διευκολύνουν το styling των component. Διαθέτει πάνω από 40 components (Button, Input etc) και παρέχει theme με ετοιμη παλέτα χρωμάτων και γραμματοσειρών για τα components αυτα. Δίνεται επίσης η δυνατότητα παραμετροποίησης του theme αυτου. Τέλος, σε κάθε component μπορεί να προστεθεί εξτρα styling , είτε μέσω του theme, είτε μεσω css/jss/emotion styles, είτε μέσω του hook που παρέχει το material ui.

```
const classes = useStyles();

return (
    <TableRow className={classes.row} {...props}>
        {children}
    </TableRow>
);
```

Εικόνα 15. Custom MUI component

2.9 VEGA LITE

Το vega lite αποτελεί μια γλώσσα υψηλού επιπέδου που επιτρέπει την ταχεία δημιουργία διαδραστικών απεικονίσεων δεδομένων. Παρέχει μια περιεκτική, δηλωτική JSON σύνταξη για την δημιουργία οπτικοποιήσεων για ανάλυση δεδομένων και παρουσιάσεων.

Ο compiler της vega-Lite δημιουργεί αυτόματα components οπτικοποίησης, συμπεριλαμβανομένων των αξόνων, επιγραφών και κλιμάκων. Καθορίζει προεπιλεγμένες τιμές για αυτά τα components, με βάση λεπτομερών σχεδιασμένων κανόνων. Αυτό έχει ως αποτέλεσμα στην γρήγορη δημιουργία οπτικοποιήσεων, καθώς και στην εύκολη επεξεργασία και προσαρμογή διαφόρων οπτικοποιήσεων. Με το vega lite, υπάρχει δυνατότητα ανάλυσης

δεδομένων, που υποστηρίζει data transformations, όπως aggregation, binning, sorting. Επιπλέον, δίνει την δυνατότητα μεταμορφώσεις απεικονίσεων (visual transformations), οπως stacking, faceting. Τέλος, το vega lite επιτρέπει συνθεσεις διαδραστικών οπτικοποιησεων σε διατάξεις.

Οι Vega-Lite προδιαγραφές (specifications) είναι JSON αντικείμενα που περιγράφουν ένα ευρείο φασμα διαδραστικών οπτικοποιήσεων. Τα βασικά δομικά στοιχεία των προδιαγραφών είναι τα εξής:

1. Το πεδίο data, στο οποίο γίνεται ο ορισμός των δεδομένων και δέχεται ποικίλες μορφές δεδομένων (json / csv / url etc.).

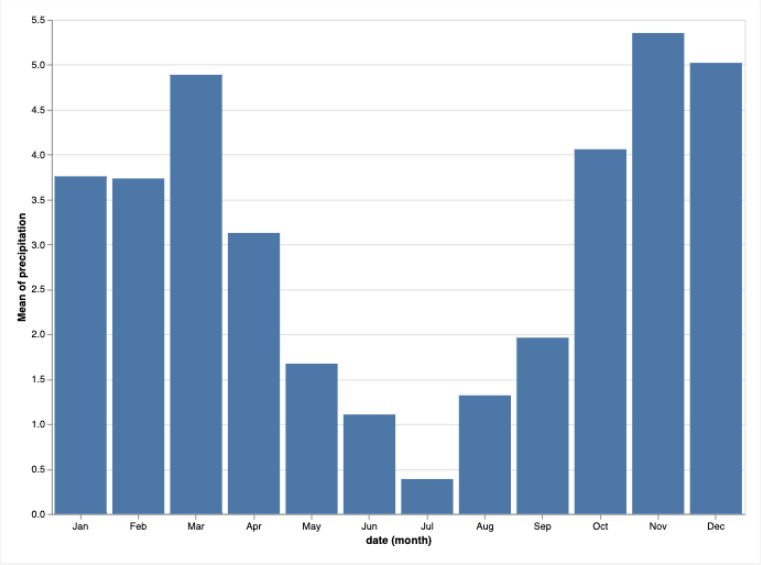
2. Το πεδίο mark, το οποίο καθορίζει τα σχήματα τα οποία θα οπτικοποιούν τα δεδομένα (πχ γραμμή, μπαρα). Σε αυτό το πεδιο μπορούν να καθοριστούν και επιπλέον χαρακτηριστικα οπως τα μεγέθοι, το χρώμα, η θέση κτλ.

3. Το πεδίο encoding, το οποίο εκπροσωπεί την απεικόνιση μεταξεί κωδικοποιήσεων καναλίων (οπως το X, Y), πεδίων δεδομένων, σταθερών τιμών. [8]

```
{
  "$schema": "https://vega.github.io/schema/vega-lite/v5.json",
  "data": {"url": "data/seattle-weather.csv"},
  "mark": "bar",
  "width":700,
  "height":500,
  "encoding": {
    "x": {"timeUnit": "month", "field": "date", "type": "ordinal"},
    "y": {"aggregate": "mean", "field": "precipitation"}
  }
}
```

Εικόνα 16. Παράδειγμα vega lite προδιαγραφής

[8] https://vega.github.io/vega-lite/



Εικόνα 17. Παράδειγμα vega lite οπτικοποίησης

2.10 NO SQL - MONGODB

Η NoSQL βάση δεδομένων αποτελεί βάση που δεν είναι σε μορφή πίνακα. Οι NoSQL βάσεις διαθέτουν ποικιλία τύπων βασισμένη στο μοντέλο δεδομένων τους. Οι κύριοι τύποι είναι document, key-value, wide-column και graph. Παρέχουν ευέλικτα σχήματα και κλιμακώνονται εύκολα με μεγάλα φορτία και ογκους δεδομένων. Οι βάσεις αυτές είναι non-relational.

Υπάρχουν πολλαπλά είδη NoSQL βάσεων όπως :

1. Document βάσεις που αποθηκεύουν τα δεδομένα σε μορφή παρόμοια με JSON αντικείμενα. Κάθε αρχειο (document) διαθέτει ζευγάρια πεδίων και τιμών. Οι τιμές μπορεί να είναι string, boolean, objects, numbers κτλ.

2. Key-value βάσεις είναι πιο απλές βάσεις που κάθε αντικείμενο έχει μόνο κλειδία και τιμές.

3. Wide-column βάσεις που αποθηκεύουν δεδομένα σε πίνακες, σειρές και στήλες.

4. Graph βάσεις που αποθηκεύουν δεδομένα σε nodes και edges.

Η χρήση της NoSQL συνιστάται όταν χρειάζεται:

1. Μεγάλος όγκος δεδομένων
2. Ανάγκη για κλιμάκωση εφαρμογής.

3. Γρήγορη και ευέλικτη ανάπτυξη.
4. Microservices ή Real time streaming.


```
_id: ObjectId("60f4173ace586b73ec2761d3")
id: "binancecoin"
symbol: "bnb"
name: "Binance Coin"
image: "https://assets.coingecko.com/coins/images/825/large/binance-coin-logo...."
current_price: 259.96
market_cap: 40165973635
market_cap_rank: 4
fully_diluted_valuation: 44324650857
total_volume: 742859234
high_24h: 263.38
low_24h: 253.97
price_change_24h: 5.67
price_change_percentage_24h: 2.22951
market_cap_change_24h: 908374283
market_cap_change_percentage_24h: 2.31388
circulating_supply: 154533651.9
total_supply: 170533651.9
max_supply: 170533651.9
ath: 564.82
ath_change_percentage: -54.15109
ath_date: "2021-05-10T07:24:17.097Z"
atl: 0.03359941
atl_change_percentage: 770640.29059
atl_date: "2017-10-19T00:00:00.000Z"
```


Εικόνα 18. Παράδειγμα NoSQL document

Η MongoDB αποτελεί δωρεαν λογισμικο ανοιχτού κώδικα διαχείρισης βάσεων δεδομένων NoSQL. Η MongoDB αποθηκεύει τα δεδομένα σε Documents. Παρέχει queries, indexing και συσσωρευσεις δεδομένων (aggregation) σε πραγματικό χρόνο, που μπορούν να χρησιμοποιηθούν για την στην πρόσβαση και στην ανάλυση των δεδομένων. Είναι κατανεμημένη βάση δεδομένων στην βάση του, γεγονός που την καθιστά ιδανική για γεωγραφική διανομή των δεδομένων, καθώς και την κλιμάκωση της εφαρμογής.[9]

[9]https://www.mongodb.com/

ΚΕΦΑΛΑΙΟ 3: Οπτικοποίηση δεδομένων

3.1 Εισαγωγή

Ανέκαθεν υπήρχε η ανάγκη για προβολή τεράστιων ποσοτήτων δεδομένων με τρόπο τέτοιο ώστε να είναι εύκολα κατανοητός και προσβάσιμος. Τα δεδομένα αυξάνονται καθημερινά. Επομένως είναι δύσκολο, ακομα και για έναν εξειδικευμένο χρήστη να μπορέσει να εξερευνήσει αυτά τα πολλαπλά συμπυκνωμένα δεδομένα. Κατα αυτον τον τρόπο, η χρήση εργαλείων οπτικοποίησης καθιστάται αναγκαία για επιστημονική, και οχι μονο, ερευνα. Με την χρήση τους, παρέχεται αποτελεσματική αναπαράσταση δεδομένων, δίνοντας την δυνατότητα στους χρήστες να δούνε αναλυτικά σε εικονική μορφή τα δεδομενα αυτα, βοηθώντας τους να αντιληφθούν τα δεδομένα και να βγάλουν τα ανάλογα πορίσματα.[10]

3.2 Τυποι ταμπλο (dashboards)

Ενα ταμπλό αποτελεί εργαλείο διαχείρισης πληροφοριών το οποίο χρησιμοποιείται για να διαχειριστεί πληροφορίες και δεδομένα. Τα ταμπλό συχωνέυουν και οπτικοποιούν δεδομένα από πολλαπλές πηγές, όπως βάσεις δεδομένων, αρχεία. Μέσω των ταμπλό, γίνεται δυνατή η παρακολούθηση διαφόρων χαρακτηριστικών και επιδοσεων παρουσιάζοντας ιστορικά δεδομένα, δεδομένα πραγματικού χρόνου και τα λοιπά. Η χρήση των ταμπλό αποτελούν σημαντικά στην χρήση τους, καθώς απλοποιούν την ανάλυση των δεδομένων, και διευκολύνουν στην κατανόηση των προβαλλόμενων δεδομένων, αποφευγοντας την υπερφόρτωση δεδομένων.

Υπάρχουν 4 βασικές υποκατηγορίες ταμπλό:

1. Τα στρατηγικά ταμπλο (strategic), τα οποία εστιάζουν σε μακροχρόνιες στρατηγικές και μετρήσεις υψηλού επιπέδου.
2. Τα λειτουργικά ταμπλό (operational), τα οποία εστιάζουν σε μικρότερα χρονικά πλαίσια και λειτουργικές διεργασίες.
3. Αναλυτικά ταμπλο (analytical), τα οποία περιέχουν τεράστιο όγκο δεδομένων από δεδομένα φτιαγμένα από αναλυτές.
4. Τακτικά ταμπλό (tactical), τα οποία χρησιμοποιούνται για την ιχνηλάτηση επίδοσης.

[10] International Journal of Engineering Research And Advanced Technology(IJERAT) ISSN: 2454-6135 [Volume. 02 Issue.12, December– 2016]

3.2.1 Στρατηγικά ταμπλό

Ένα στρατηγικό ταμπλό είναι ένα εργαλείο που χρησιμοποιείται για την παρακολούθηση μακροχρόνιων στρατηγικών και την ιχνηλάτηση επιδόσεων. Ως αποτέλεσμα, αυτά τα ταμπλό τείνουν να συνοψίζουν τα δεδομένα χρονικά, όπως για παράδειγμα σε μήνες ή χρονια. Αποτελούν τα πιο πολύπλοκα και δύσκολα ταμπλό στην κατανόηση.

3.2.2 Λειτουργικά ταμπλό

Ένα λειτουργικό ταμπλό αποτελεί εργαλείο το οποίο χρησιμοποιείται για την παρακολούθηση και διαχείριση διεργασιών που έχουν μικρότερο χρονικό εύρος. Αποτελούν τα πιο συνηθισμένα και απλά ταμπλό, καθώς και τα πιο εύκολα στην κατανόηση.

3.2.3 Αναλυτικά ταμπλό

Αναλυτικά ταμπλο (analytical), τα οποία περιέχουν τεράστιο όγκο δεδομένων από δεδομένα φτιαγμένα από αναλυτές. Προσφέρουν στον χρήστη μια περιεκτική ανασκόπηση των δεδομένων, και στοχεύουν σε μεσοπρόθεσμη ανάλυση. Αυτά τα ταμπλό συνήθως συνεβρίσκονται με τα λειτουργικά ή τα στρατηγικά ταμπλό, και δεν είναι τόσο πολύπλοκα όσο τα στρατηγικά ταμπλό.

3.2.4 Τακτικά ταμπλό

Τακτικά ταμπλό (tactical), τα οποία χρησιμοποιούνται για την ιχνηλάτηση επίδοσης. Αποτελούν τα πιό αναλυτικά ταμπλό, και διαθέτουν διαφορα φίλτρα για την ξενάγηση στα δεδομένα, και στοχεύουν σε μεσοπρόθεσμη ανάλυση. Επιπλέον, διαθέτουν περισσότερες απεικονίσεις δεδομένων. Αυτά τα ταμπλό συνήθως συνεβρίσκονται με τα λειτουργικά ή τα στρατηγικά ταμπλό, και δεν είναι τόσο πολύπλοκα όσο τα στρατηγικά ταμπλό.

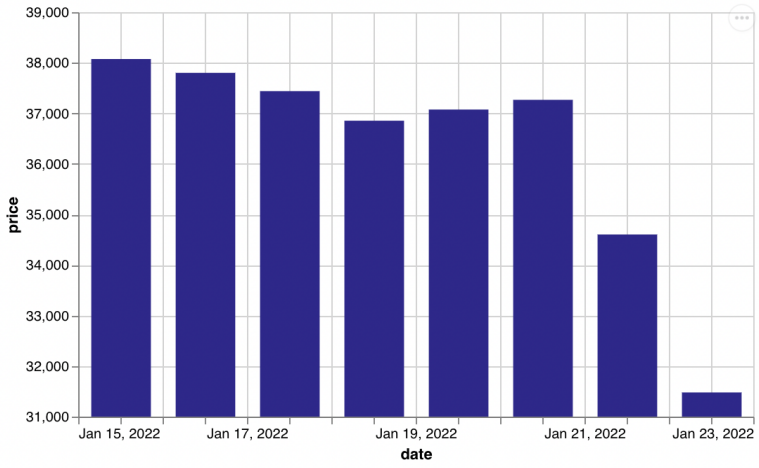
3.3 Τύποι γραφημάτων που χρησιμοποιήθηκαν

Η οπτικοποίηση των δεδομένων είναι δυνατή με πολλαπλούς τρόπους, κάθε μία από τις οποίες έχουν συγκεκριμένο σκοπό και χαρακτηριστικές διαφορές. Τα διαγράμματα που χρησιμοποιήθηκαν στην παρούσα εργασία είναι κυρίως τα εξής:

1. Bar chart
2. Stacked bar charts
3. Grouped bar charts
4. Line chart
5. Area chart
6. Candlestick chart

3.3.1 Bar chart

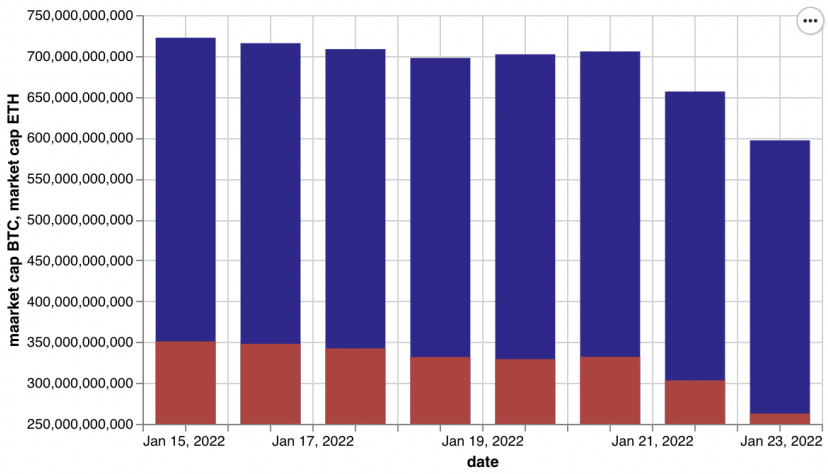
Τα charts αυτά είναι τα πιο συνηθισμένα για την οπτικοποίηση δεδομένων, κυρίως γιατί είναι γρήγορο να συγκρίνεις πληροφορίες και να ξεχωρίσεις αλλαγές στις τιμές με ευκολία. Τα ραβδογράμματα είναι ιδιαίτερα αποτελεσματικά όταν θέλουμε να χωρίσουμε αριθμητικά δεδομένα σε διάφορες κατηγορίες. Όταν πρόκειται για χρονικές συχνότητες, τότε το γράφημα ονομάζεται histogram.



Εικόνα 19. Bar chart

3.3.2 stacked bar chart

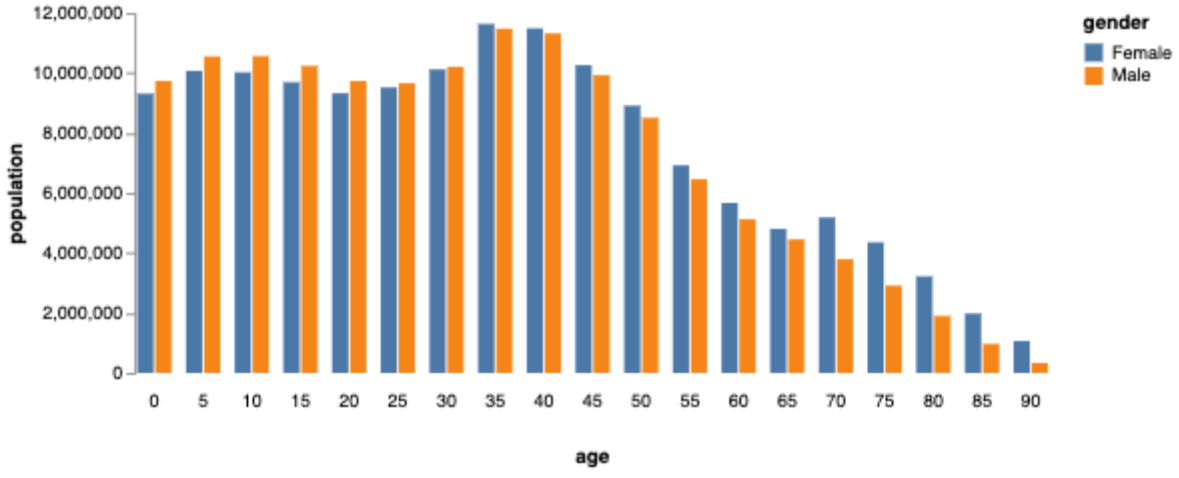
Το stacked bar chart είναι σαν ένα απλό bar chart, στο οποίο όμως η κάθε μπάρα έχει χωριστεί σε μικρότερες μπάρες, κάθε μία από τις οποίες σηματοδοτεί ένα στοιχείο διαφορετικής κατηγορίας στα δεδομένα.



Εικόνα 20. Stacked Bar chart

3.3.3 Grouped bar chart

Το grouped bar chart επεκτείνει ένα απλό ραβδόγραμμα, με την διαφορά ότι συγκρίνονται δίπλα δίπλα δεδομένα με διαφορετικές κατηγορίες.



Εικόνα 21. Grouped Bar chart

3.3.4 Line chart

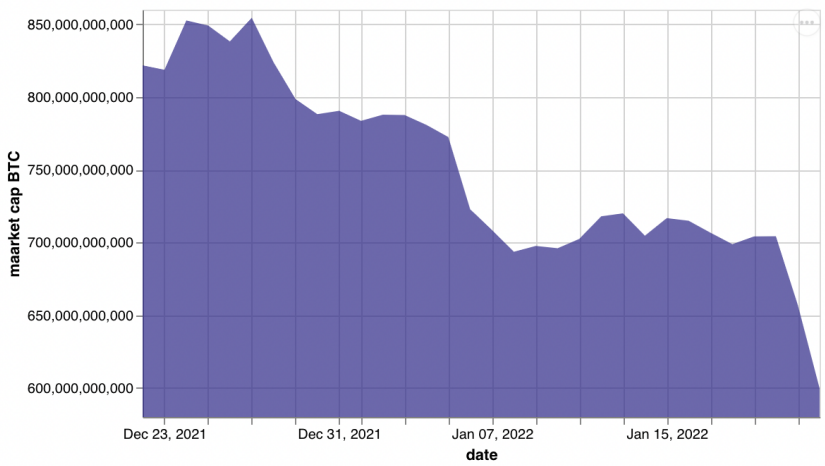
Αυτό το διάγραμμα χρησιμοποιείται κυρίως για να δείχνει την αλλαγή διαφόρων τιμών σε σχέση με τον χρόνο. Κάθε σημείο τοποθετείται από τα αριστερά στα δεξιά, με την κάθετη θέση να ορίζει την τιμή του σημείου αυτού. Τελος, τα σημεία αυτα ενώνονται με γραμμές για να δώσουν έμφαση στην πρόοδο του χρόνου.[11]

Εικόνα 22. Line chart

3.3.5 Area chart

Αυτό το διάγραμμα είναι ουσιαστικά ένα Line διαγραμμα, στο οποίο όμως το κενό μεταξύ της γραμμής και του άξονα X είναι χρωματισμένο με χρώμα ή κάποιο μοτίβο.[12] ‘Οταν πρόκειται για γραφήματα με πολλαπλές οντότητες, τότε στο διάγραμμα χρειάζεται ένας βαθμός διαφάνειας, καθώς οι γραμμές μπορεί να αλληλεπικαλύπτονται.

[11] How to Choose the Right Data Visualization Chartio
[12] Data Visualization 101: How to Choose the Right Chart or Graph for Your Data



Εικόνα 23. Area chart

3.3.6 Candlestick chart

Το Διαγράμματα Κηροπήγιο αποτελεί γράφημα το οποίο χρησιμοποιείται για τη λήψη αποφάσεων σε συναλλαγές μετοχών, συναλλάγματος, εμπορευμάτων και δικαιωμάτων προαίρεσης. Βλέποντας ένα κηροπήγιο, μπορεί κανείς να αναγνωρίσει τις τιμές ανοίγματος και κλεισίματος ενός στοιχείου, τα υψηλά και τα χαμηλά, καθώς και το συνολικό εύρος για ένα συγκεκριμένο χρονικό πλαίσιο.[13] Κάθε "κηροπήγιο" απεικονίζει μια συγκεκριμένη χρονική περίοδο. Για παράδειγμα, ένα διάγραμμα ενός μήνα που δημιουργεί ένα κηροπήγιο ανα ημέρα, θα έχει συνολικά 30 κηροπήγια.

Στο γράφημα αυτό, πρέπει να έχουμε δεδομένα που περιέχουν τις τιμές open, high, low και close, για κάθε περίοδο που θέλουμε να απεικονίσουμε. Το πιο έντονα χρωματισμένο σημείο ονομάζεται σώμα (body). Αποτελείται από δύο μέρη, το άνοιγμα και το κλείσιμο(open / close). Αν το κλείσιμο είναι μεγαλύτερο από το άνοιγμα, τότε το κηροπήγιο αποκτά χρώμα που απεικονίζει άυξηση (συνήθως πράσινο). Κατα αυτόν τον τρόπο, το άνοιγμα βρίσκεται στο κάτω μέρος του σώματος, ενώ το κλείσιμο στο πάνω. Αντιθέτως, οταν το κλείσιμο είναι μικρότερο, τότε αποκτά χρώμα που απεικονίζει μείωση (συνήθω κόκκινο). Κατα αυτόν τον τρόπο, το άνοιγμα βρίσκεται στο πάνω μέρος του σώματος, ενώ το κλείσιμο στο κάτω. Οι λεπτές γραμμές πάνω και κάτω από το σώμα εκπροσωπούν τις ελάχιστες και μέγιστες τιμές της χρονικής περιόδου (high/low) και ονομάζονται μέγιστα και ελάχιστα ("wicks" και "tails").

[13] Lu, Tsung-Hsun; Shiu, Yung-Ming; Liu, Tsung-Chi (2012-04-01). "Profitable candlestick trading strategies—The evidence from a new perspective". *Review of Financial Economics*. 21 (2): 63–68.

Εικόνα 24. Candlestick chart

ΚΕΦΑΛΑΙΟ 4: Σχεδίαση και αρχιτεκτονική

4.1 Εισαγωγή

Η λειτουργία της εφαρμογής βασίζεται σε ένα nodeJS Restful API, το οποίο διαθέτει διάφορα endpoints, τα οποία καθιστούν δυνατή την επικοινωνία του front end με την βάση δεδομένων. Η βάση αυτή διαθέτει δεδομένα τα οποία προέρχονται από ένα ανοιχτό - δημόσιο API κρυπτονομισμάτων (Public API). Το δημόσιο αυτο API, καλείται ανα τακτά χρονικά διαστήματα προσφέροντας μας φρέσκα δεδομένα κρυπτονομισμάτων, τα οποία συλλέγονται στην βάση δεδομένων.

Εικόνα 25: Σχεδιαγραμμα οντοτητων

4.2 OPEN API

Ένα ανοιχτό ή δημόσιο API, αποτελεί μία εφαρμογή που είναι δημόσια διαθέσιμη στους προγραμματιστές. Είναι δημοσιευμένα στο διαδίκτυο και η χρήση τους είναι δωρεάν. Παρέχουν δεδομένα και υπηρεσίες. Συνήθως είναι REST API, και ή απόκτηση των δεδομένων γίνεται δυνατή μέσω RESTful μεθόδων (GET, POST etc.), και η ανταπόκριση έρχεται σε μορφή JSON.

Η εφαρμογή χρησιμοποιεί δεδομένα που προέρχονται από το ανοιχτό API της εφαρμογής CoinGecko , το οποίο παρέχει δωρεάν δεδομένα κρυπτονομισμάτων. Η CoinGecko αποτελεί εφαρμογή που παρέχει εργαλεία και δεδομένα για την παρακολούθηση των τιμών, την κεφαλαιοποίηση της αγοράς και άλλα στατιστικά στοιχεία.

Το ανοιχτό API της εφαρμογής CoinGecko παρέχει αρκετά endpoints τα οποία μπορούν να χρησιμοποιηθούν για στατιστική ανάλυση. Συγκεκριμένα, έγινε επιλογή του endpoint που

φέρνει στατιστικά στοιχεία για την συγκεκριμένη χρονική στιγμή, μέσω του endpoint: /coins/markets που δεχεται πολλαπλά id κρυπτονομισμάτων που μας ενδιαφέρουν (πχ bitcoin, ada, etherium etc). Καλώντας αυτό το Endpoint με μέθοδο GET, έρχεται response το οποίο διαθέτει χρήσιμα δεδομένα για τα κρυπτονομίσματα που έχουμε συμπεριλάβει. Στην περιπτωση της εφαρμογής, ζητάω τα στοιχεία 10 κρυπτονομισμάτων (BinanceCoin,cardano,etherium, polkadot, bitcoin, dogecoin, litecoin, ripple, tether, usd-coin)


```
{
  "id": "bitcoin",
  "symbol": "btc",
  "name": "Bitcoin",
  "image": "https://assets.coingecko.com/coins/images/1/large/bitcoin.png?1547033579",
  "current_price": 48107,
  "market_cap": 908439441458,
  "market_cap_rank": 1,
  "fully_diluted_valuation": 1010251479051,
  "total_volume": 36799205222,
  "high_24h": 53038,
  "low_24h": 47489,
  "price_change_24h": -4209.757075129434,
  "price_change_percentage_24h": -8.04664,
  "market_cap_change_24h": -79456311995.69604,
  "market_cap_change_percentage_24h": -8.04299,
  "circulating_supply": 18883643,
  "total_supply": 21000000,
  "max_supply": 21000000,
  "ath": 59717,
  "ath_change_percentage": -19.01949,
  "ath_date": "2021-11-10T14:24:11.849Z",
  "atl": 51.3,
  "atl_change_percentage": 94169.96812,
```


Εικόνα 26. CoinGecko Response

4.3 Συλλογή δεδομένων

Η συλλογή δεδομένων γίνεται δυνατή με την χρήση NodeJS. Κάθε 10 λεπτά, καλείται μία ασυγχρονη συνάρτηση, η οποία ζητάει δεδομένα από το ανοιχτό API, και εφόσον πάρει το response, το αποθηκεύει στην βάση δεδομένων.

Για αρχή, αυτό υλοποιήθηκε με την μέθοδο setInterval, για οσο τρέχει ο nodeJS server, και αργότερα αυτή η λογική μεταφέρθηκε στο cloud, οπου διαθέτει διάφορα εργαλεία για πυροδότηση μεθόδων σε βάθος χρόνου όπως για παράδειγμα το google cloud scheduler. Στην συγκεκριμένη περίπτωση, χρησιμοποιήθηκε ο heroku cloud scheduler, ο οποίος πυροδοτεί μια διεργασία (μια ασύγχρονη javascript μέθοδο που κανει GET τα δεδομένα από το coinGecko και ύστερα τα αποθηκεύει στην βάση) κάθε 10 λεπτά.

Χάρη στον cloud scheduler, μειώνεται η πιθανότητα σφαλμάτων, και τρέχει με χρονική ακρίβεια, χωρίς να επηρεάζεται από τον κεντρικό back end server σε περίπτωση που πεσει.

4.4 Βάση δεδομένων

Για την συλλογή δεδομένων, είναι απαραίτητη η πρόσβαση σε μια βάση δεδομένων. Εφόσον έρθουν τα δεδομένα, αποθηκεύονται επιτόπου στην βάση δεδομένων noSQL (mongoDB). Μια βάση δεδομένων noSQL αποτελείται από collections, ενώ ενα collection απο documents, τα οποία συνήθως αποτελούν JSON objects. Κάθε ένα από τα νομίσματα αποτελεί ένα collection, και τα στατιστικά δεδομένα κάθε χρονικής στιγμής (που έρχονται σε μορφή JSON) αποτελούν ένα document. Επομένως, η βάση αποτελείται από 10 collections :

1. BinanceCoin
2. Cardano
3. Etherium
4. Polkadot
5. Bitcoin
6. Dogecoin
7. Litecoin
8. Ripple
9. Tether
10. Usd-coin

Κάθε ένα από αυτά τα collections διαθέτουν πολλά documents τα οποία αποτελούν τις τιμές του κάθε κρυπτονομίσματος για καθε χρονική στιγμή (ανα δεκα λεπτα).

Collection Name	Documents	Documents Size	Documents Avg	Indexes	Index Size	Index Avg
binancecoin	12168	8.82MB	761B	1	268KB	268KB
bitcoin	12168	8.31MB	716B	1	272KB	272KB
cardano	12168	8.62MB	743B	1	268KB	268KB
dogecoin	12168	8.35MB	720B	1	268KB	268KB
ethereum	12168	9.08MB	783B	1	272KB	272KB
litecoin	12168	8.48MB	731B	1	272KB	272KB
polkadot	12168	8.57MB	739B	1	268KB	268KB
ripple	12168	8.72MB	752B	1	280KB	280KB
tether	12168	8.48MB	731B	1	272KB	272KB
usd-coin	12168	8.54MB	736B	1	276KB	276KB

Εικόνα 27: collections

Κάθε document αποθηκεύεται κάθε 10 λεπτά σε κάθε συλλογή, κρατόντας τα στατιστικά στοιχεία της συγκεκριμένης χρονικής περιόδου. Πολλαπλά documents θα χρησιμοποιηθούν μετά για την οπτικοποίηση των δεδομένων κάθε κρυπτονομίσματος.


```
_id: ObjectId("60f416fdce586b73ec2761c6")
id: "bitcoin"
symbol: "btc"
name: "Bitcoin"
image: "https://assets.coingecko.com/coins/images/1/large/bitcoin.png?15470335..."
current_price: 26988
market_cap: 506019216190
market_cap_rank: 1
fully_diluted_valuati…: 566446607371
total_volume: 14866544252
high_24h: 27533
low_24h: 26640
price_change_24h: 298.02
price_change_percenta…: 1.1166
market_cap_change_24h: 5472515058
market_cap_change_per…: 1.09331
circulating_supply: 18759762
total_supply: 21000000
max_supply: 21000000
ath: 54205
ath_change_percentage: -50.41891
ath_date: "2021-04-14T11:54:46.763Z"
atl: 51.3
atl_change_percentage: 52290.33113
atl_date: "2013-07-05T00:00:00.000Z"
roi: null
last_updated: "2021-07-18T11:56:36.404Z"
date: 2021-07-18T11:00:00.000+00:00
```


Εικόνα 28: documents

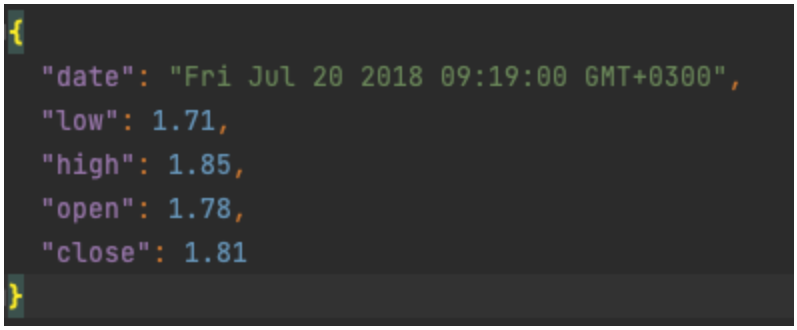
4.5 BACK END API

Η back end εφαρμογή έχει φτιαχτεί με nodeJS. Το API που έχει φτιαχτεί αποτελείται από 3 endpoints, απο τα οποία το καθένα έχει τον δικό του σκοπό. Τα endpoint είναι τα εξής :

1. **/getByIdCustom:** Αυτό το endpoint προορίζεται για τα απλά διαγράμματα τιμών κρυπτονομισμάτων (price, market cap, etc), και καλείται με την GET μεθοδο. Παίρνει 4 query παραμέτρους, το ID του κρυπτονομίσματος, την συχνότητα (daily, weekly, monthly, 6-months, yearly), fields (πεδία που μας ενδιαφέρουν, οπως current price. Default: ολα) και date, το οποίο προσδιορίζει από πότε θέλουμε να πάρουμε δεδομένα.

Ανάλογα με αυτές τις παραμέτρους, διαλέγουμε και τις αντίστοιχες τιμές απο την βάση δεδομένων. Για παράδειγμα, αν έχουμε id bitcoin, daily συχνότητα και date το σημερινό, τότε θα μας φέρει όλες τις δεκάλεπτες εγγραφές της σημερινής ημέρας, για το bitcoin, ταξινομημένες.

2. **/getCandleStickDataById:** Αυτό το endpoint προορίζεται για το διάγραμμα κηροπήγιο, που απαιτεί, ανοιγματα, κλεισιματα, μεγιστες και ελάχιστες τιμές ανα χρονική περίοδο (open, close, min, max). Υπολογίζει αυτές τις τιμές με βάση τα δεδομένα που υπάρχουν στην βάση δεδομένων. Καλείται με την GET μεθοδο. Παίρνει 4 query παραμέτρους, το ID του κρυπτονομίσματος, την συχνότητα (daily, weekly, monthly), interval (ωρες που θέλουμε να υπολογίσουμε τις μεγιστες και ελάχιστες τιμές.) και date, το οποίο προσδιορίζει από πότε θέλουμε να πάρουμε δεδομένα. Εφόσον έρθουν τα δεδομένα από την βάση, τότε γίνεται επεξεργασία. Αναλογα με το interval, γίνεται εύρεση των μέγιστων και των ελαχίστων τιμών στα δεδομένα που έχουν επιστραφεί απο την βάση δεδομένων. Για παράδειγμα, αν το interval είναι 12, και η συχνότητα είναι weekly, θα έρθουν απο την βάση όλες οι τιμές της εβδομάδας. Τότε, με βάση αυτά τα δεδομένα, θα βρεί ανα 12ωρο τις τιμές min max, και οι τιμές open close θα είναι η τιμή της αρχής του 12ωρου και το τέλος αντίστοιχα. Τελος, επιστρέφει μια λίστα από αντικείμενα με πεδία date, low, high, open, close.



Εικόνα 29: Response object getCandleStickDataById

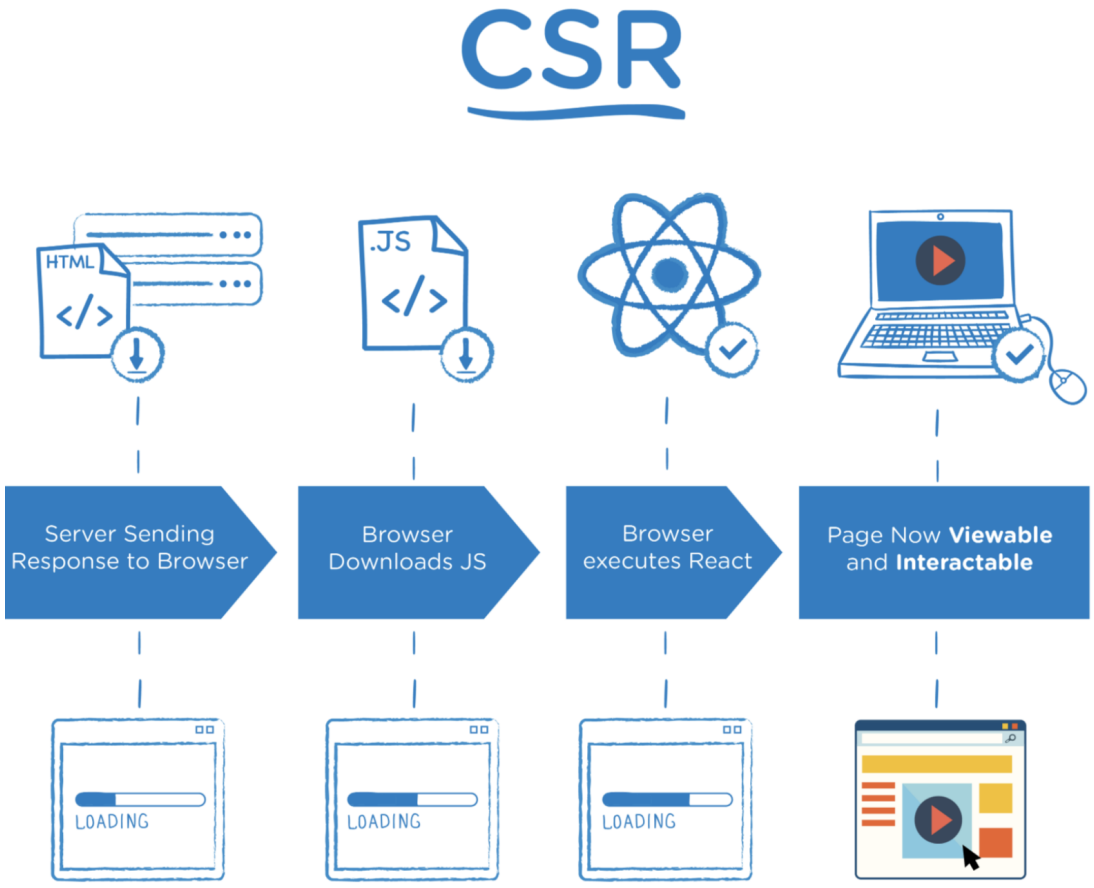
3. **/getOverviewStats:** Αυτό το endpoint προορίζεται για τα ταμπλό που συνοψίζουν τις τιμές των κρυπτονομισμάτων, και δεν χρησιμοποιούνται σε κάποιο διάγραμμα. Δέχονται σαν παραμέτρους πολλαπλά id κρυπτονομισμάτων, και date. Επιστρέφει πίσω ολα τα στατιστικά της ημέρας, όπως την τιμή, market cap, total volume, min 24 hours, max 24 hours, all time high, weekly price percentage change και άλλα στατιστικά.

4.6 FRONT END SPA APPLICATION

Στην εφαρμογή αυτή, γίνονται οι παρουσιάσεις των δεδομένων. Εδώ, γίνεται η απεικόνιση των δεδομένων σε μορφή διαγραμμάτων και ταμπλό.

Με την React, υπάρχει δυνατότητα να δημιουργηθούν εφαρμογές που φορτώνονται στον server και έπειτα στέλνονται στον χρήστη (Server Side Rendering), ή να φορτώνονται στον χρήστη (Client Side Rendering). Οι εφαρμογές αυτές λεγονται SPA (single page application).

Μία εφαρμογή SPA αποτελεί μια δικτυακή εφαρμογή η οποία αλληλεπιδρά με τον χρήστη ξαναγράφοντας δυναμικά την υπάρχουσα σελίδα με νέα δεδομένα από τον server, αντί να φορτώνει ολόκληρες σελίδες από την αρχή. Αυτό οδηγεί στην γρηγορότερες μεταβάσεις, κάνοντας την σελίδα να έχει την αίσθηση native εφαρμογής.



Εικόνα 30: Client side rendering (CSR)

Όταν ο χρήστης κάνει αίτημα στον server, τότε επιστρέφεται ένα κενό HTML αρχείο και ένα μεγάλο bundle με JavaScript. Όταν εκτελεστεί αυτό το bundle, τότε εκτελεί διάφορες API κλήσεις, και γεμίζει την σελίδα με το html, τα styles και τα δεδομένα που χρειάζεται. Αυτή η μέθοδος είναι πιό αργή κατα την διάρκεια της πρώτης φόρτωσης του bandle από πλευράς χρήστη, καθώς όλες οι διαδικασίες για την αιτήματα συμβαίνουν στον υπολογιστή του.

ΚΕΦΑΛΑΙΟ 5: Υλοποίηση και κώδικας

5.1 Εισαγωγή

Η πτυχιακή αυτή αποτελείται από 2 εφαρμογές, οι οποίες έχουν γραφτεί σε javascript και typescript. Η πρώτη είναι το BACK END API, το οποίο παίρνει τα δεδομένα από την βάση και τα στέλνει στο FRONT END, η δεύτερη είναι ή React εφαρμογή.

5.2 Rest API

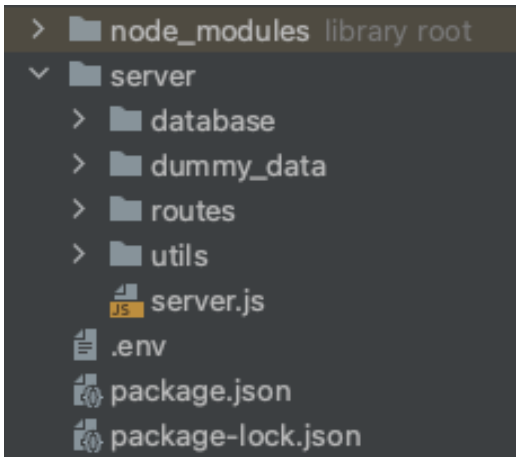
Το Rest API έχει υλοποιηθεί με nodeJS, σε γλώσσα Javascript. Κύρια χρήση του είναι η λήψη δεδομένων από την βάση, και η αποστολή τών δεδομένων αυτών στην React εφαρμογή μέσω των ακόλουθων Rest endpoints.

URL	**QUERY PARAMS**	**HTTP VERB**	**RESULT**
/api/getByIdCustom	{id, date, frequency, fields}	GET	Array of coin data, size depending on frequency.
/api/getOverviewStats	{coins, date}	GET	Returns Array of the coin's overview stats. Each array item represents a coin.
/api/getCandleStickDataById	{id, date, frequency, interval}	GET	Returns an array of objects used for candlestick charts (min, max, open, close).

Πινακας 1. REST endpoints

5.2.1 Μορφολογία φακέλων και αρχεία

Οι 2 βασικοί φάκελοι της εφαρμογής είναι το node_modules, και το server.



Εικόνα 31: nodeJS directories

1. Το node_modules, αποτελεί φάκελος στον οποίο αποθηκεύονται όλα τα πακέτα που εγκαθίστανται μέσω του NPM, για να χρησιμοποιηθούν στην εφαρμογή. Τα περιεχόμενα του φακέλου αυτού εξαρτόνται από το package.json, το οποίο αποτελεί την καρδιά όλης της εφαρμογής, καθώς εκεί αναγράφονται τα πακέτα και οι εκδόσεις που θα εγκατασταθούν στα node_modules, καθώς και πληροφορίες για την εφαρμογή, scripts για την εκκίνηση της κτλ.

```
{
  "name": "opengraphapi",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "app": "node server/server.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "axios": "^0.21.1",
    "cors": "^2.8.5",
    "dotenv": "^10.0.0",
    "express": "^4.17.1",
    "moment": "^2.29.1",
    "mongodb": "^4.0.0",
    "now": "^21.0.1"
  }
}
```

Εικόνα 32: package.json

2. Ο φάκελος server περιέχει όλους τους φακέλους στους οποίους υπάρχουν όλα τα javascript αρχεία.

A. Στο αρχείο server.js, στο οποίο γίνεται η αρχικοποίηση της δέσμευσης και ακοής του server.

```
const express = require('express');
const apiRouter = require('./routes')
const app = express()

const PORT = process.env.PORT || 8080
app.use(express.json())
app.use('/api', apiRouter)

app.listen(PORT, hostname: () => {
    console.log('successfully deployed')
})
```

Εικόνα 33: server.js

B. Ο φάκελος database διαθέτει όλο τον κώδικα για την σύνδεση και απολαβή των δεδομένων από την βάση δεδομένων.

```
const mongoCredentials = process.env['MONGO_CREDENTIALS']
const client = new MongoClient(mongoCredentials);

async function main() {
    try {
        await client.connect();
    } catch (e) {
        console.error(e, 'ERROR');
    }
}
main().catch(console.error)

const mongodb = {}
mongodb.getSingleCoinCustom = async (coinId, frequency, date, fields) => {...}
mongodb.getCandleStickData = async (coinId, frequency, interval, date) => {...}
mongodb.getOverviewStats = async (coinsSelected :any[] = [], date) => {...}
module.exports = mongodb
```

Εικόνα 34: mongo.js

C. Ο φάκελος dummy_data διαθέτει αρχεία json και js τα οποία περιέχουν τις διάφορες επιλογές για τις παραμέτρους που δέχονται τα endpoints, όπως το frequency.json, το οποίο έχει μέσα τις τιμές daily, weekly, monthly etc.

D. Ο φάκελος utils διαθέτει διάφορα javascript functions τα οποία χρησιμοποιούνται για την προετοιμασία δεδομένων.

```
const preparePercentage = (startPrice, endPrice) => {
    if (startPrice && endPrice) {
        return (100 - (startPrice * 100 / endPrice))
    }
    return 0
}

module.exports = {preparePercentage}
```

Εικόνα 35: utils -> preparePercentage.js

Ε. Ο φάκελος routes διαθέτει το αρχείο στο οποίο βρίσκονται οι αρχικοποιήσεις των endpoints, και το κάλεσμα των μεθόδων για την επικοινωνία και την ανάκτηση δεδομένων από την βάση.

```
const express = require('express')
const router = express.Router()
const mongodb = require('../database/mongo')
const cors = require('cors')
const {mongoFieldsAll, mongoFields} = require('../dummy_data/mongoFieldNames')

router.get( path: '/getByIdCustom', cors(), async (request : Request<P, ResBody, ReqBody, ReqQuery, Locals> , response : Response<any, Record<string, any>> , next : NextFunction ) => {...

router.get( path: '/getCandleStickDataById', cors(), async (request : Request<P, ResBody, ReqBody, ReqQuery, Locals> , response : Response<any, Record<string, any>> , next : NextFunction
    try {
        const id = request.query?.id
        const frequency = request.query?.frequency
        const interval = request.query?.interval
        const date = request.query?.date
        const res = await mongodb.getCandleStickData(id, frequency, interval, date)
        return response.json(res)
    } catch (e) {
        console.log(e)
        return response.status( code: 500).send(e)
    }
})

router.get( path: '/getOverviewStats', cors(), async (request : Request<P, ResBody, ReqBody, ReqQuery, Locals> , response : Response<any, Record<string, any>> , next : NextFunction ) => {


module.exports = router
```

Εικόνα 36: routes -> index.js

5.3 Frontend εφαρμογή

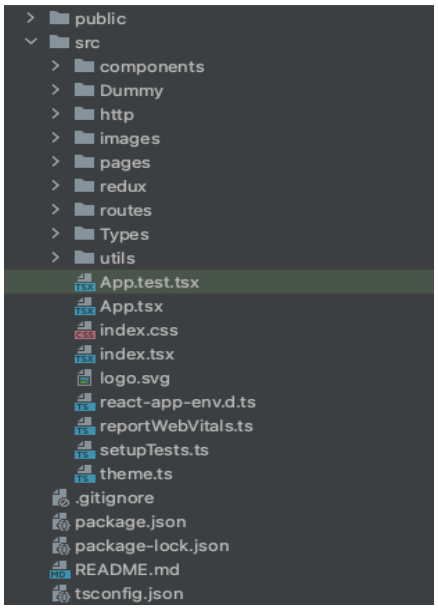
Η front end εφαρμογή έχει υλοποιηθεί με την γλώσσα typescript, χρησιμοποιώντας την βιβλιοθήκη ReactJs. Σε αυτήν την εφαρμογή γίνεται η παρουσίαση των δεδομένων, και δημιουργία των διαδραστικών διαγραμμάτων.

Η αρχικοποίηση της εφαρμογής γίνεται με την εντολή npx create-react-app createApp, η οποία δημιουργεί τα απαραίτητα αρχεία, εγκαθιστά βασικά πακέτα και κάνει τις καταλληλες παραμετροποιήσεις για την δημιουργία του κατάλληλου οικοσυστήματος. Παραπάνω βιβλιοθήκες που χρησιμοποιήθηκαν είναι:

- Material UI, το οποίο παρέχει μια μεγάλη συλλογή έτοιμων components
- VegaLite, το οποίο χρησιμοποιείται για την οπτικοποίηση των δεδομένων σε διαγράμματα
- Axios, το οποίο χρησιμοποιείται για τα http Requests
- Redux-toolkit, που αποτελεί εργαλείο διαχείρισης δεδομένων (state management tool), και χρησιμοποιείται για την αποθήκευση των προσαρμοσμένων διαγραμμάτων

5.3.1 Μορφολογία φακέλων και αρχεία

Η μορφολογία των φακέλων είναι η εξής.



Εικόνα 37: Μορφολογία ReactJs εφαρμογής

1. Ο φάκελος Components διαθέτει όλα τα React Components που χρησιμοποιούν στην εφαρμογή, όπως διάφορα Inputs, cards, dialogs κτλ.
2. Στον φάκελο dummy υπάρχουν αρχεία json που περιέχουν δεδομένα που χρησιμοποιούνται για διάφορες λειτουργίες της εφαρμογής, όπως οι ονομασίες των νομισμάτων, διάφορες vega lite επιλογές, κτλ.
3. Ο φάκελος http, διαθέτει τα αρχεία στα οποία γίνονται τα GET request και η αρχικοποίηση του axios.
4. Ο φάκελος pages διαθέτει όλες τις σελίδες της εφαρμογής.
5. Ο φάκελος redux διαθέτει το redux store, και τα ανάλογα slices.
6. Ο φάκελος Types διαθέτει τους τύπους των http responses, των vega lite επιλογων .
7. Ο φάκελος utils διαθέτει διάφορες συναρτήσεις για προετοιμασία δεδομένων (όπως τον υπολογισμό ποσοστιαίας μεταβολής).

Βασικά αρχεία για την ReactJS εφαρμογή αποτελούν:

1. Τα αρχεία για τα πακέτα και για τις ρυθμίσεις της typescript και άλλων παραμέτρων (package.json, tsconfig.json etc).
2. Ο φάκελος public, περιέχει το html αρχείο της εφαρμογής. Μέσα σε αυτό το αρχείο υπάρχει το στοιχείο με id root, το οποίο θα χρησιμοποιηθεί σαν container για το reactDOM
3. Ο φάκελος src, ο οποίος περιέχει τα αρχεία και φακέλους που χρησιμοποιουνται στην react εφαρμογή. Βασικοί φάκελοι και αρχεία αποτελούν:
    a. Index.tsx, στο οποίο υπάρχει η συνάρτηση που καθιστά δυνατή την εμφάνιση των ReactJS στοιχείων. Το ReactDOM.render δέχεται σαν παραμέτρους κάποιο react component, και το html στοιχείο που θα χρησιμοποιηθεί σαν container για το react component.

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';

ReactDOM.render(
    <App/>,
    document.getElementById( elementId: 'root')
);
```

Εικόνα 38: index.tsx

b. Το app.tsx αποτελεί component το οποίο διαθέτει αρκετά providers που θα χρησιμοποιηθούν στην εφαρμογή. Συγκεκριμένα, το provider της redux, το Material UI provider, το οποίο παρέχει διάφορα στυλ για τα react Components, τον BrowserRouter, που είναι υπεύθυνο για το routing της εφαρμογής, και το layout, το οποίο ορίζει την διάταξη της σελίδας.

```
function App() {
    return (
        <Provider store={store}>
            <PersistGate loading={null} persistor={persistor}>
                <MuiThemeProvider theme={theme}>
                    <BrowserRouter>
                        <Layout>
                            <Routes/>
                        </Layout>
                    </BrowserRouter>
                </MuiThemeProvider>
            </PersistGate>
        </Provider>
    );
}

export default App;
```

Εικόνα 39: App.tsx

4. Το Routes.tsx διαθέτει τα urls που επικρατούν στην εφαρμογή. Για τον ορισμό των paths, χρησιμοποιείται η βιβλιοθήκη react-router. Το component Route δεχεται σαν prop το path, και το prop component δέχεται το Component που θα δείξει στο path αυτό.

```
const Routes: FC = () => {
    return (
        <Switch>
            <Route exact path={'/create'} component={CreatePage}/>
            <Route exact path={'/compare'} component={CompareCoinPage}/>
            <Route exact path={'/'} component={CoinPreviewPage}/>
            <Route exact path={'/preview'} component={CoinPreviewPage}/>
            <Route exact path={'/preview/:id?'} component={CoinSelectedPage}/>
        </Switch>
    );
}

export default Routes;
```

Εικόνα 40:Routes.tsx

5.3.2 Προδιαγραφές VEGA LITE

Για να γίνει η εισαγωγή του vega Lite στην σελίδα, χρησιμοποιείται η βιβλιοθήκη vega-embed. Η βιβλιοθήκη αυτή διαθέτει την συνάρτηση vegaEmbed, η οποία δέχεται τις εξής παραμέτρους, το στοιχείο στο οποίο θέλουμε να γίνει η οπτικοποίηση, και το vega spec, το οποίο είναι τύπου JSON. Για κάθε διάγραμμα, χρησιμοποιείται αυτό το Component, απλώς με διαφορετικό vegaConfig σαν prop.

Η useEffect πυροδοτεί την συνάρτηση κάθε φορα πού το vega Config αλλάζει. Το vegaEmbed δέχεται σαν πρώτη παράμετρο το id του element στο οποίο θέλουμε να απεικονίσουμε την οπτικοποίηση, και σαν δεύτερη παράμετρο το vega lite αντικείμενο. Το vegaEmbed επιστρέφει μια υπόσχεση (Promise), την οποία μπορούμε να διαχειριστούμε, σε περίπτωση που υπάρξει κάποιο σφάλμα.

```
const VegaLitePreview: FC<VegaComponentProps & BoxProps> = ({className, vegaConfig, keyId : string = 'idkey', ...props : ... }) => {
    const classes = useStyles();

    useEffect( effect: () => {
        vegaEmbed( el: `#${keyId}`, vegaConfig).then(() => console.log('success')).catch((e) => console.log('e'));
    }, deps: [vegaConfig])

    return (
        <div className={`${classes.root} ${className}`} id={keyId} {...props}/>
    );
}

export default VegaLitePreview;
```

Εικόνα 41:VegaLite component

Τα βασικά διαγράμματα αποτελούν components, τα οποία έχουν σαν state κάποιο vega spec, το οποίο περνάει σαν prop στο παραπάνω Component. Το vegaConfig prop δέχεται το JSON vega lite αντικείμενο, το οποίο έχει προετοιμαστεί ανάλογα με το διάγραμμα που θέλουμε να απεικονίσουμε.

```
<Grid item xs={12}>
    <VegaLitePreview
        style={{width: '100%', background: '#f0f0f0', boxShadow: '0 0 0 2px #72621d', borderRadius: 4}}
        vegaConfig={vega} keyId={`vega-comp-${field}`}/>
</Grid>
```

Εικόνα 42: Χρήση του VegaLite component

Στο Vega lite αντικείμενο ορίζονται όλες οι ιδιότητες της οπτικοποίησης. Για αρχή ορίζονται το background (HEX), το width, το height. Έπειτα, ο ορισμός των δεδομένων γίνεται στο πεδίο data.

```
width: "container",
height: "container",
background: '#f0f0f0',
data: {values: dataArray},
```

Εικόνα 43: Ορισμός δεδομένων και διαμέτρων

Μετά ορίζεται ο τύπος του διαγράμματος, δηλαδή άμα είναι ραβδογράφημα, γραμμής κτλ. Το πεδίο υπεύθυνο για αυτό είναι το mark.

```
mark: {
    type: "area",
    width: '',
    line: true, "point": hasPoints,
    tooltip: true,
    opacity: 0.1
},
```

Εικόνα 44: Ορισμός τύπου διαγράμματος

Μετά ορίζεται ο άξονας x και y, ποιά πεδία απεικονίζουν και την επεξεργασία των δεδομένων. Συγκεκριμένα αυτές οι δηλώσεις γίνονται μέσα στο encoding πεδίο, το οποίο δέχεται ένα αντικείμενο με πεδία x / y. Μέσα σε αυτά τα πεδία μπορούν να οριστούν ο τύπος του δεδομένου, να γίνει επεξεργασία (πχ aggregate), και να οριστούν και καποια διακοσμητικά στοιχεία που σχετίζονται με τον άξονα.

```
encoding: {
    x: {
        title: "",
        field: "date",
        type: "temporal",
        timeUnit: "yearmonthdatehoursminutes",
        axis: {
            gridDash: [5, 5],
            labelColor: "#02254b",
            titleColor: "#02254b"
        }
    },
    y: {
        aggregate: "average",
        stack: "none",
        scale: {"zero": false},
        field: "customField",
        title: `${field}`,
        type: "quantitative",
        axis: {
            gridColor: "lightgray",
            labelColor: "#02254b",
            titleColor: "#02254b"
        }
    },
```

Εικόνα 45: Ορισμός άξονα x και y

Στην δημιουργία των προσαρμοσμένων διαγραμμάτων, η vega προδιαγραφή έχει μοιραστεί σε διαφορετικές μεταβλητές. Συγκεκριμένα, υπάρχουν μεταβλητές για:

1. Τον άξονα X
2. Τον άξονα Y
3. Γενικές προδιαγραφές του vega
4. Την ένδειξη (τυπο του διαγράμματος)
5. Τα δεδομένα

Όλες αυτές οι μεταβλητές, συγκεντρώνονται μεσα σε μια νέα μεταβλητή, η οποία αποτελεί το τελικό vega lite specification. Κάθε φορά που αλλάζει κάποια απο αυτές τις μεταβλητές, τότε η μεταβλητή για το specification ανανεώνεται χάρη στην useEffect.

```
useEffect( effect: () => {
    setVlSpec( value: {
        data: {
            values: [...coinData]
        },
        ...simpleStyles,
        ...transform,
        ...mark,
        encoding: {
            y: {...yAxis},
            x: {...xAxis},
        }
    })
}, deps: [coinData, simpleStyles, transform, xAxis, yAxis, mark])
```

Εικόνα 46:Δημιουργία προσαρμοσμένης προδιαγραφής

Στο τέλος, η νέα αυτή προδιαγραφή περνάει σαν prop στο component το οποίο έχει φτιαχτεί για να απεικονίζει τις vega προδιαγραφές.

ΚΕΦΑΛΑΙΟ 6: Σενάρια χρήσης

Στην εφαρμογή υπάρχουν οπτικοποιήσεις και ταμπλο που δίνουν στον χρήστη την δυνατότητα παρατήρησης, συσχέτισης και σύγκρισης δεδομένων κρυπτονομισμάτων, οδηγώντας τον στην καλύτερη κατανόηση των δεδομένων αυτών. Σε αυτήν την εφαρμογή, μέσω ταμπλό και διαγραμμάτων δίνεται η δυνατότητα:

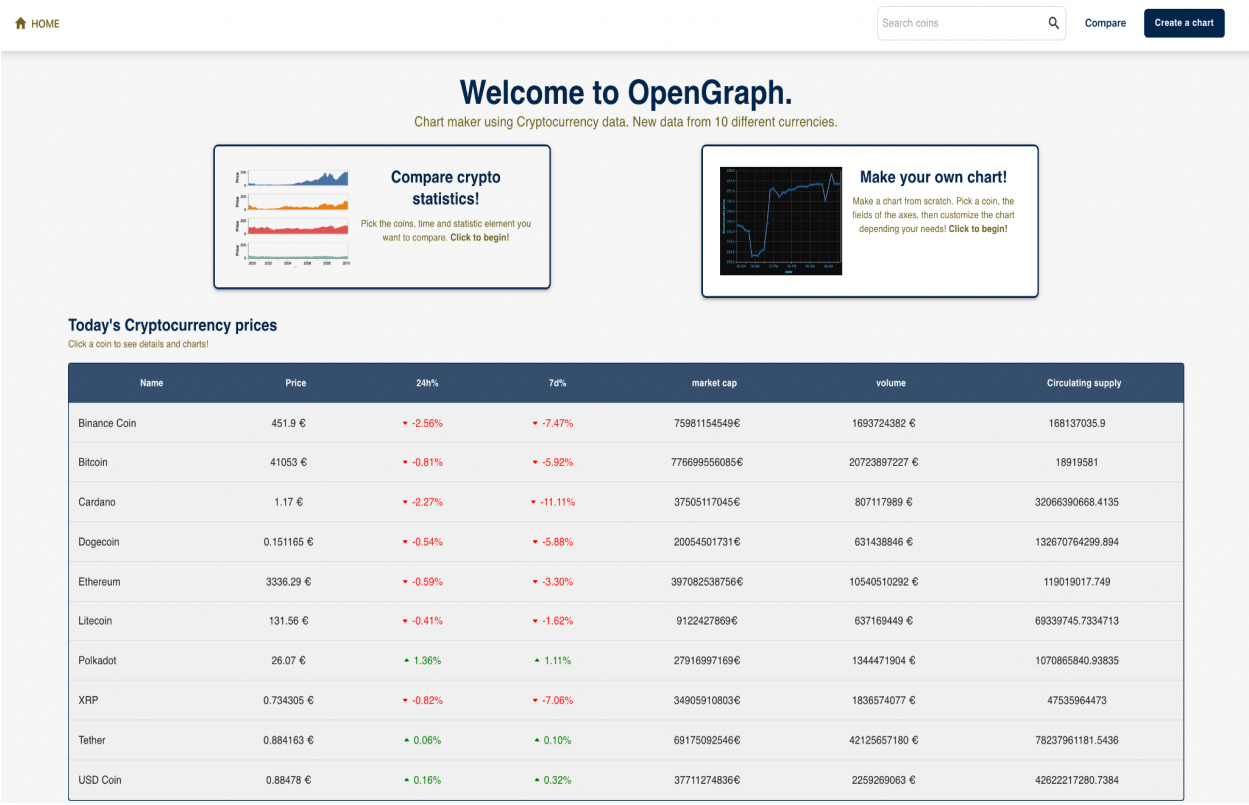
- Αναλυτικής οπτικοποίησης δεδομένων ενός κρυπτονομίσματος.
- Σύγκρισης δεδομένων πολλαπλών κρυπτονομισμάτων.
- Δημιουργία διαγραμμάτων με βάση τις προτιμήσεις του χρήστη.

6.1 Αρχική σελίδα

Στην αρχική σελίδα γίνεται η εμφάνιση βασικών οντοτήτων που βοηθούν στην πλοήγηση στην εφαρμογή. Στην επικεφαλίδα, υπάρχουν συντομεύσεις για διάφορα τμήματα της σελίδας. Στο περιεχόμενο της σελίδας υπάρχουν δύο βασικές καρτέλες που μας οδηγούν στα αντίστοιχα τμήματα της εφαρμογής.

- Στην αριστερή καρτέλα, θα οδηγηθούμε στην σελίδα σύγκρισης δεδομένων κρυπτονομισμάτων.
- Στην δεξιά καρτέλα, θα οδηγηθούμε στην σελίδα δημιουργίας προσαρμοσμένης οπτικοποίησης.

Στον πίνακα, υπάρχουν σύντομες πληροφορίες για τα κρυπτονομίσματα της εφαρμογής, ο οποίος έχει την δυνατότητα ταξινόμησης για κάθε πεδίο. Τέλος, αν πατήσουμε σε μία από τις γραμμές, θα πάμε στο τμήμα της εφαρμογής που μας παρουσιάζει εκτενής πληροφορίες για το συγκεκριμένο νόμισμα.



Εικόνα 47:Αρχική σελίδα

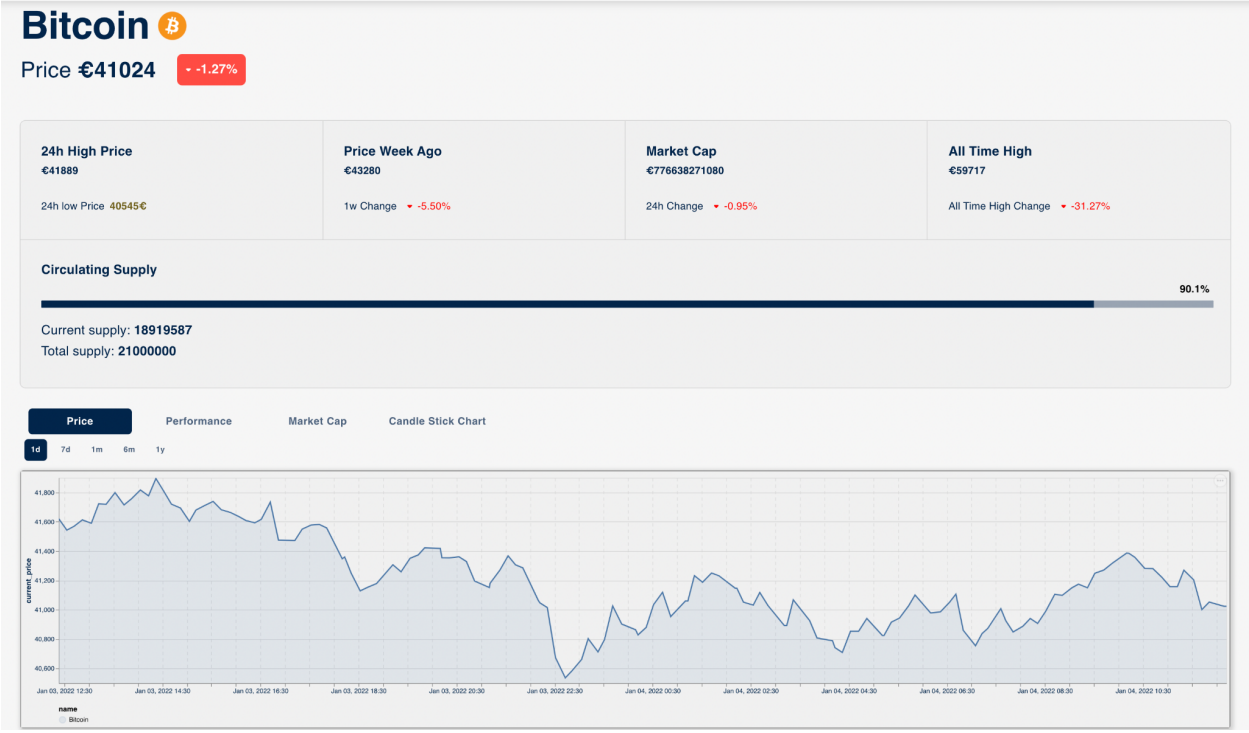
6.2 Σελίδα πληροφοριών νομίσματος

Σε αυτήν την σελίδα ερχόμαστε όταν πατήσουμε σε ένα στοιχείο του πίνακα ή απο την μπάρα αναζήτησης. Εδώ, υπάρχουν αναλυτικά πολλά δεδομένα ενός κρυπτονομίσματος. Συγκεκριμένα, βλέπουμε ένα ταμπλό που απεικονίζει:

- την τιμή
- την ποσοστιαια μεταβολη της ανα 24 ωρες, μια εβδομάδα
- την ελάχιστη και μέγιστη τιμή
- την συνολική αξία στην αγορά
- το υπάρχων και το τελικό πλήθος των νομισμάτων.

Τέλος, υπάρχουν διαγράμματα γραμμής, που απεικονίζουν ιστορικές τιμές - χρηματιστηριακή αξία και ποσοστιαία μεταβολής της τιμής σε βάθος χρόνου με δυνατότητα ορισμού του εύρους του χρόνου σε 1 μέρα, 1 εβδομάδα 1 μήνα, 6 μήνες και 1 χρόνο. Επιπλέον υπάρχει το διάγραμμα Κηροπήγιο, που απεικονίζει τιμές ανοιγματος, κλεισιματος, μεγιστων, ελαχιστων. Στο διάγραμμα αυτό μπορούμε να ελέγξουμε το χρονικο περιθώριο ημερών μέσω του φίλτρου, και να κάνουμε μεγέθυνση στο γράφημα μέσω την βοηθητικής μπάρας από κάτω.

Κατα αυτόν τον τρόπο, ο χρήστης μπορεί να κατανοήσει την πορεία του νομίσματος σε βάθος χρόνου, και να λάβει ανάλογες αποφάσεις σε περίπτωση που θέλει να επενδύσει.



Εικόνα 48: Σελίδα πληροφοριών νομίσματος

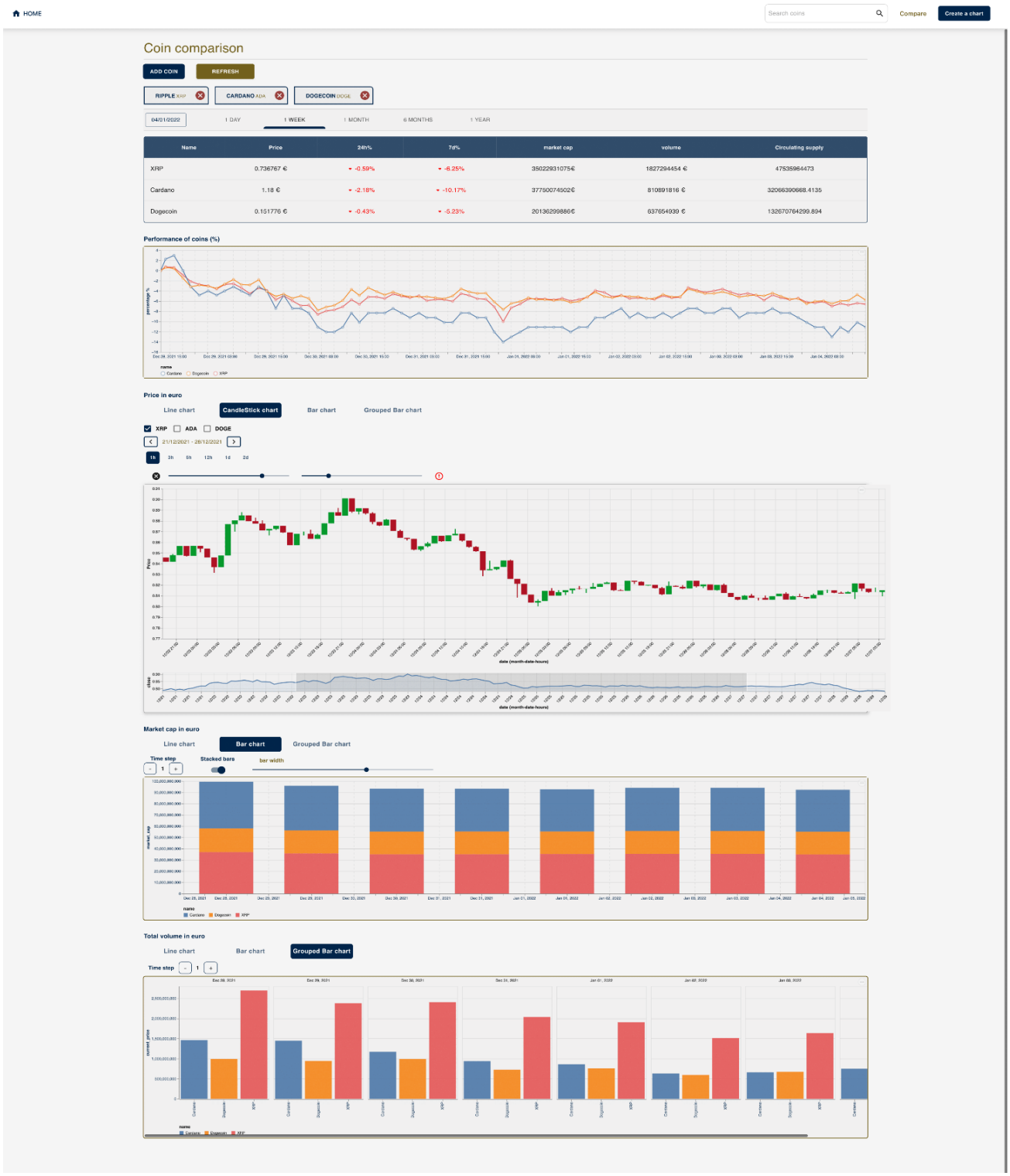
Εικόνα 49: Σελίδα πληροφοριών νομίσματος και γράφημα κηροπήγιο

6.3 Σελιδα συγκρισης νομισμάτων

Σε αυτήν την σελίδα γίνεται η σύγκριση πολλαπλών κρυπτονομισμάτων. Με την επιλογή ADD COIN ανοίγει ένα παράθυρο που μας επιτρέπει να διαλέξουμε πολλαπλά νομίσματα για σύγκριση. Εφόσον γίνει η επιλογή, τότε μας φέρνει δεδομένα για τα νομίσματα αυτά, για την τελευταία εβδομάδα. Το χρονικό εύρος μπορεί να αλλάξει, καθώς υπάρχουν και να αντίστοιχα φίλτρα (ημερομηνία, και χρονικό ευρος πχ 1 εβδομαδα, 1 μήνας κτλ).
Τα διαγράμματα που απεικονίζονται είναι τα εξής:

- Το πρώτο διάγραμμα απεικονίζει την ποσοστιαία μεταβολή της τιμής των κρυπτονομισμάτων, για το συγκεκριμένο χρονικό ευρος. Κάθε γραμμή αποτελεί και την ποσοστιαία μεταβολή ενός κρυπτονομίσματος, και ξεχωρίζει από το χρώμα του.
- Το δεύτερο διάγραμμα απεικονίζει την τιμή σε βάθος χρόνου. Η μορφή του σε Διαγράμματος αλλάζει με τις καρτέλες.
    1. Στην πρώτη καρτέλα τα νομίσματα εμφανίζονται στο διάγραμμα σαν γραμμές, η κάθε γραμμή με διαφορετικό χρώμα.
    2. Στην δεύτερη καρτέλα, κάθε τιμή νομίσματος εμφανίζεται στο διαγραμμα με κηροπήγια.
    3. Στο τρίτο διάγραμμα, οι τιμές των νομισμάτων εμφανίζονται με μπάρες, τοποθετημένες η μία πάνω από την άλλη (stack bar chart).
    4. η τέταρτη καρτέλα, εμφανίζει τις τιμές χωρισμένες σε μπάρες, δίπλα δίπλα (grouped bar chart) και κάθε ομάδα εκπροσωπεί μία συγκεκριμένη χρονική περίοδο (πχ μια μέρα).
- Στο δεύτερο διάγραμμα απεικονίζεται η χρηματιστηριακή αξία.Αντίστοιχα, και εδώ υπάρχουν καρτέλες για διαγράμματα γραμμής, stacked και grouped ραβδογραμματα.
- Στο τρίτο διάγραμμα απεικονίζεται το σύνολο συναλλαγών του κάθε νομίσματος. Αντίστοιχα, και εδώ υπάρχουν καρτέλες για διαγράμματα γραμμής, stacked και grouped ραβδογραμματα.

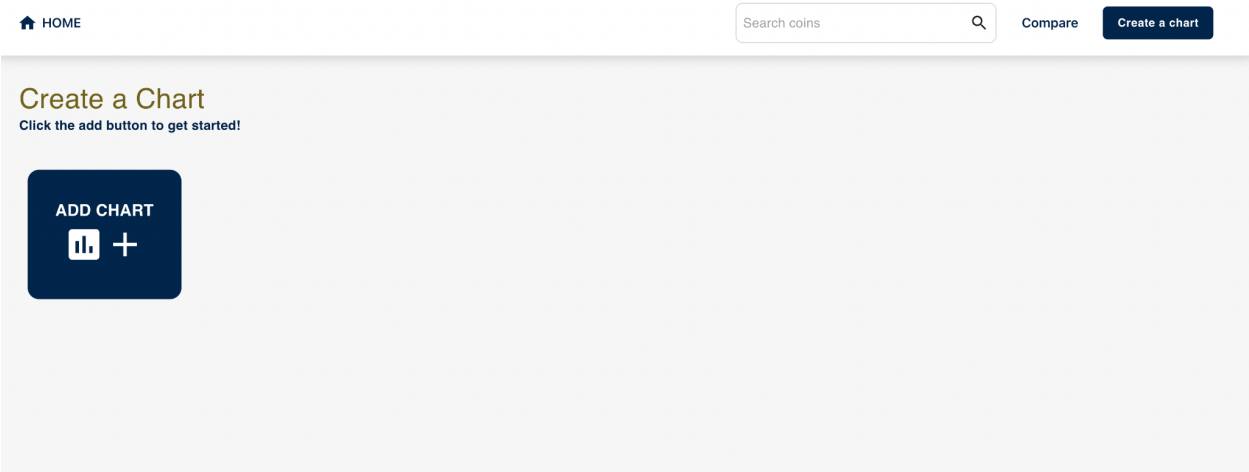
Σε κάθε διάγραμμα που διαθέτει μπάρες, δίνεται η δυνατότητα παραμετροποίησης των μεγεθών της.



Εικόνα 50: Σελίδα σύγκρισης νομισμάτων

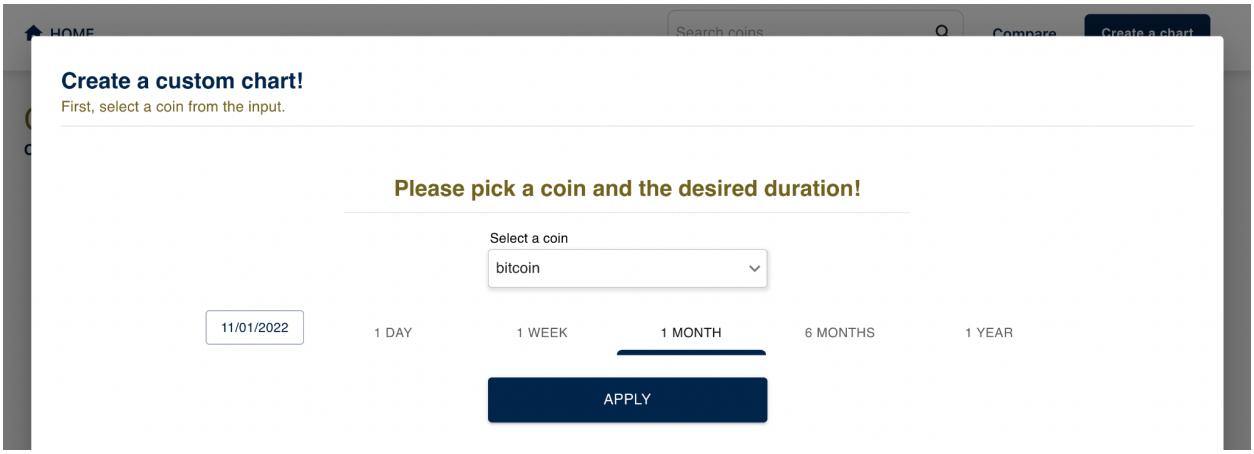
6.4 Σελίδα δημιουργίας διαγραμμάτων

Σε αυτή την σελίδα γίνεται η δημιουργία παραμετροποιημένου διαγράμματος. Πατώντας το κουμπί ADD CHART ανοίγει το παράθυρο δημιουργίας.



Εικόνα 51: Σελίδα δημιουργίας γραφημάτων

Στο πρώτο παράθυρο γίνεται η επιλογή του νομίσματος και η χρονική περίοδος. Με το κουμπί Apply προχωράμε στο παράθυρο δημιουργίας του γραφήματος.



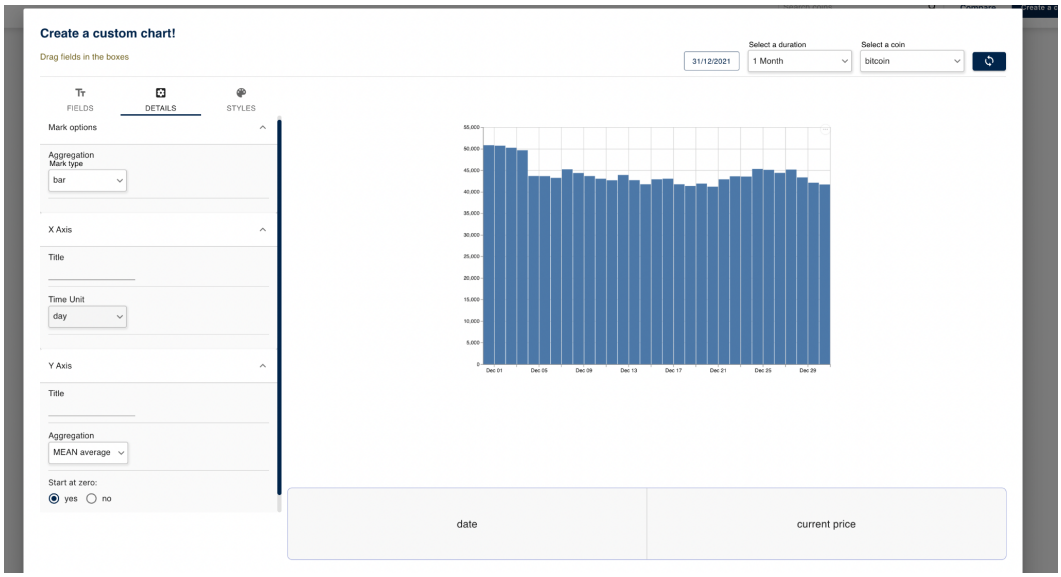
Εικόνα 52. Επιλογή δεδομένων για το διάγραμμα

Αυτό το παράθυρο αποτελείται από 3 καρτέλες δημιουργίας, το διάγραμμα, και τα φίλτρα που έχουμε επιλέξει για τα δεδομένα.

- Η πρώτη καρτέλα διαθέτει τα πεδία των δεδομένων. Για να διαλέξουμε ένα πεδίο, πρέπει να το σύρουμε στα τετράγωνα X και Y, που βρίσκονται κάτω από το διάγραμμα.

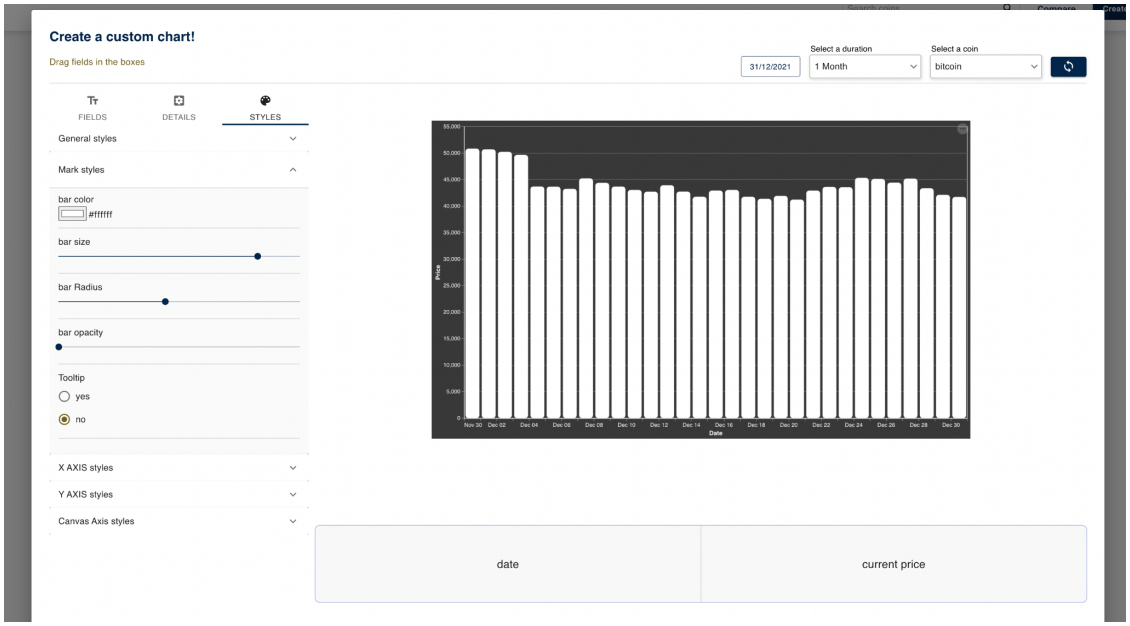
Εικόνα 53: Επιλογή πεδίων για το διάγραμμα

- Η δεύτερη καρτέλα διαθέτει τις επιλογές για
  - τον τύπο του διαγράμματος(bar, line, square, circle, tick, area, point, rule)
  - Τον άξονα X και τον άξονα Y, όπως τον τίτλο, και ανάλογα με τον τύπο των δεδομένων του άξονα (αν είναι ημερομηνία ή αριθμος), τότε δίνει επιλογές για συναρτήσεις συναθροίσεων.



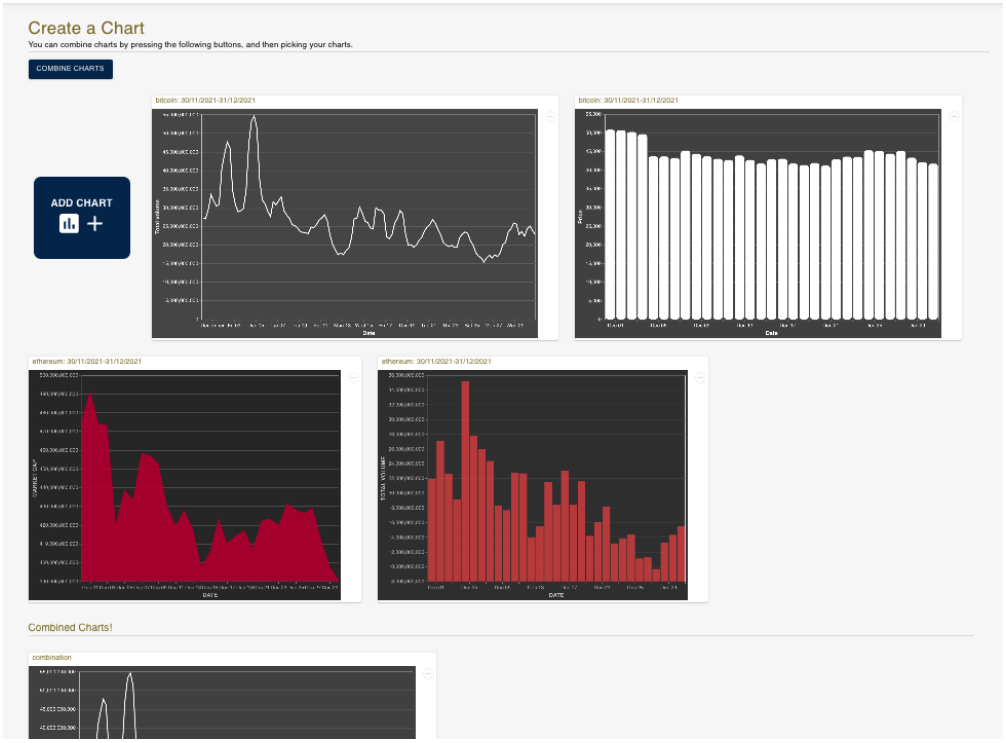
Εικόνα 54: Επιλογή ρυθμίσεων δεδομένων για το διάγραμμα

- Η τρίτη καρτέλα εστιάζει στο στύλ του διαγράμματος. Διαθέτει
    - General styles, οπου γίνεται παραμετροποίηση του στυλ για ολο το διάγραμμα, όπως μέγεθος και χρώμα.
    - Mark styles, για το στυλ του σημαδιού (χρώμα, μέγεθος κτλ).
    - X axis & Y axis, οπου ρυθμίζονται οι γραμμές και το χρώμα της γραμμής του αξονα.
    - Canvas Axis Styles, για τα χρώματα των τίτλων/γραμμών του καμβα.



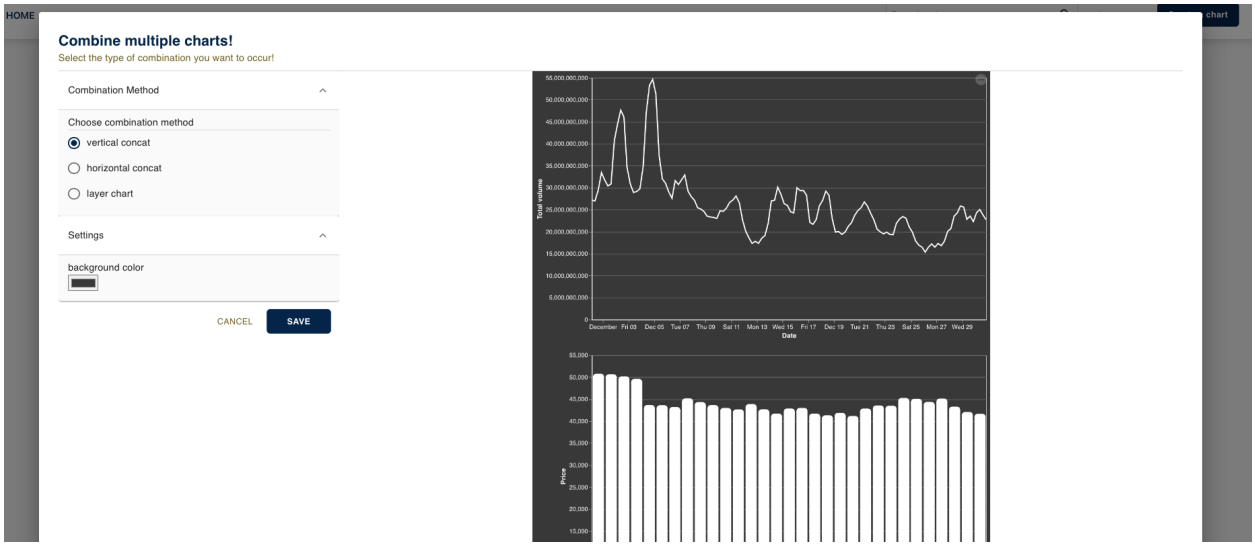
Εικόνα 55: Επιλογή στυλ για το διάγραμμα

Η αποθήκευση του διαγράμματος γίνεται με το κουμπί αποθήκευση. Έπειτα, επιστρέφουμε στην αρχική οθόνη που εμφανίζονται τα διαγράμματα που έχουν φτιαχτεί. Όλα τα διαγράμματα αποθηκεύονται στο local storage του browser. Σε αυτήν την σελίδα έπειτα μπορούμε να προσθέσουμε παραπάνω διαγράμματα, να τα επεξεργαστούμε, και να τα συνδυάσουμε. Για να γίνει ο συνδυασμός, πατάμε το κουμπί COMBINE, και έπειτα διαλέγουμε τα δυο διαγράμματα.

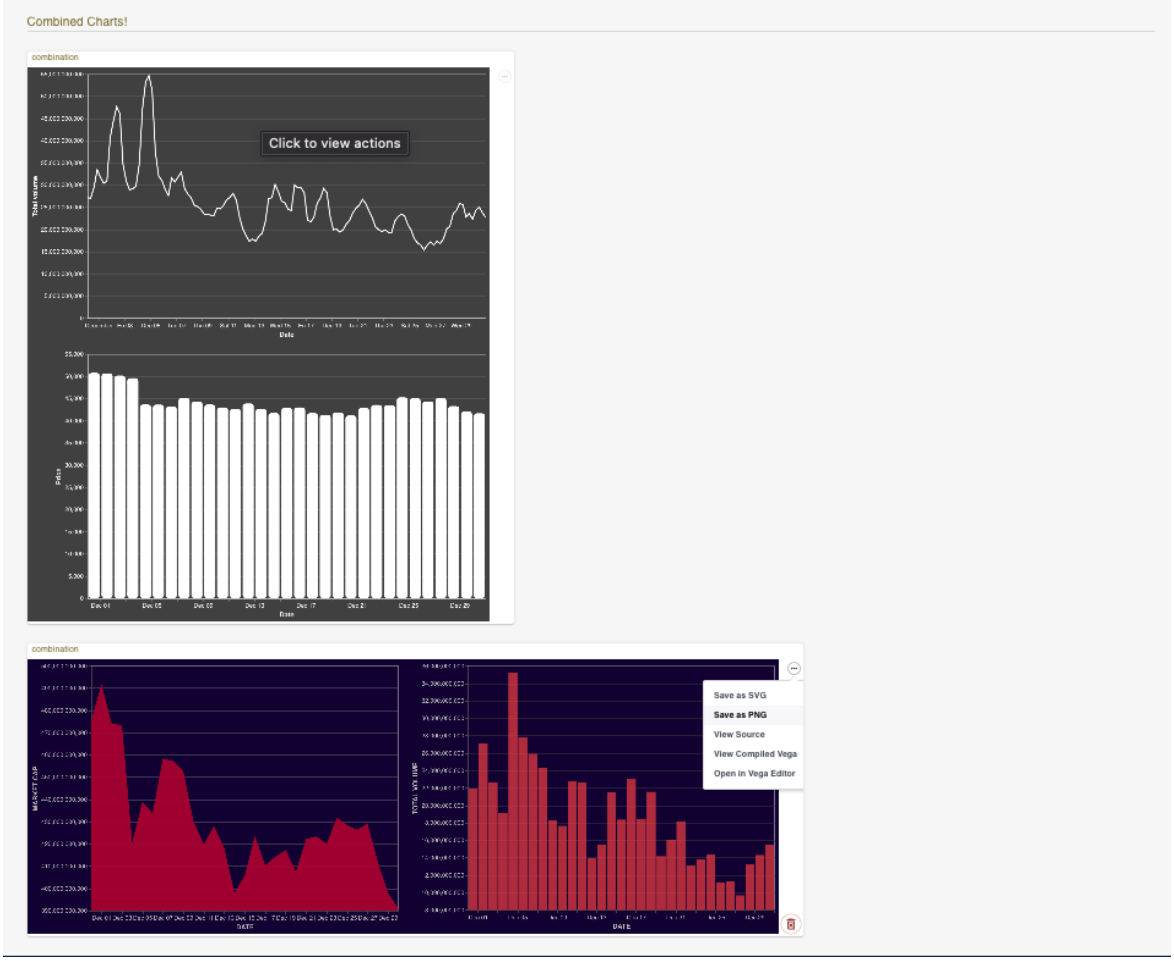


Εικόνα 56: Πολλαπλά διαγράμματα

Υπάρχει η δυνατότητα να ενωθούν δύο διαγράμματα κάθετα, οριζόντια και να ενωθούν στο ίδιο διάγραμμα. Τα αποτελέσματα της ένωσης χωρίζονται σε καινούριο τομέα.



Εικόνα 57: Ένωση διαγραμμάτων



Εικόνα 58: Αποτελέσματα ένωσης

ΚΕΦΑΛΑΙΟ 7: Συμπεράσματα

Με το πέρας αυτής της πτυχιακής εργασίας, αναπτύχθηκαν δυο εφαρμογές που κάνουν δυνατή την σχεδίαση διαδραστικών ταμπλό απο ανοιχτο API. Δημιουργήθηκε μια πλήρως λειτουργική front-end εφαρμογή, στην οποία γίνεται η οπτικοποίηση και η παραμετροποιημένη σχεδίαση των δεδομένων, και ένα back-end Restful API, το οποίο ευθύνεται για την προετοιμασία τους. Μετά από την δημιουργία αυτών των εφαρμογών, προσκομίστηκε μεγάλος όγκος γνώσεων που σχετίζονται με το web development, τοσο στο back end, οσο και στο front end και αποτέλεσε μία βασική εισαγωγή στην επιστήμη των δεδομένων.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. HTML & CSS Design and Build Websites Jon Duckett John Wiley & Sons, Inc.

2. An Analysis of the Dynamic Behavior of JavaScript Programs Gregor Richards Sylvain Lebresne Brian Burg Jan Vitek S3 Lab, Department of Computer Science, Purdue University, West Lafayette, IN {gkrichar,slebresn,bburg,jv}@cs.purdue.edu

3. Tilkov, S.; Vinoski, S. (2010). Node.js: Using JavaScript to Build High-Performance Network Programs. , 14(6), 80–83. doi:10.1109/mic.2010.145

4. International Journal of Engineering Research And Advanced Technology(IJERAT) ISSN: 2454-6135 [Volume. 02 Issue.12, December– 2016]

5. How to Choose the Right Data Visualization Chartio

6. Data Visualization 101: How to Choose the Right Chart or Graph for Your Data

7. Lu, Tsung-Hsun; Shiu, Yung-Ming; Liu, Tsung-Chi (2012-04-01). "Profitable candlestick trading strategies—The evidence from a new perspective". *Review of Financial Economics*. 21 (2): 63–68.

Πηγές

1. https://en.wikipedia.org/wiki/CSS

2. https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs#what_is_react

3. https://www.typescriptlang.org/

4. https://reactjs.org/

5. https://vega.github.io/vega-lite/

6. https://www.mongodb.com/