ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Χρηματοοικονομικός Αναλυτής - Full stack web Εφαρμογή με React & Spring Boot** **Financial Assets Analyst - Full stack Application with React & Spring Boot**
Ονοματεπώνυμο Φοιτητή	**Χατζηαθανασίου Φανούριος**
Πατρώνυμο	**Γεώργιος**
Αριθμός Μητρώου	**ΜΠΠΛ 20090**
Επιβλέπων	**Ευθύμιος Αλέπης, Αναπληρωτής Καθηγητής**

Ημερομηνία Παράδοσης **Νοέμβριος 2022**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Ευθύμιος Αλέπης Αναπληρωτής Καθηγητής	Μαρία Βίρβου Καθηγήτρια	Ευάγγελος Σακκόπουλος Αναπληρωτής Καθηγητής

Περιεχόμενα

Περίληψη.

Τα χρηματοοικονομικά μέσα λόγω της ραγδαίας εξέλιξης της τεχνολογίας, αποτελούν ένα πλέον προσβάσιμο τρόπο επένδυσης του χρήματος. Εξυπηρετεί στη ροή του χρήματος και με τη κατάλληλη έρευνα στις αγορές, μπορεί να αποφέρει κέρδη στους επενδυτές. Σκοπός αυτής της πτυχιακής εργασίας είναι η ανάπτυξη μιας Full Stack Web Application, με όνομα Financial Assets Analyst. Οι τεχνολογίες που αξιοποιήθηκαν στο backend περιλαμβάνουν Spring boot, MongoDB και στο frontend κυρίως Typescript(NextJS-React). Μέσω αυτής της εφαρμογής δίνεται η δυνατότητα στους χρήστες να έχουν πρόσβαση σε χρηματοοικονομικά γραφήματα και να χρησιμοποιούν δείκτες τεχνικής ανάλυσης, να κατεβάζουν σε μορφή CSV τα δεδομένα αυτά καθώς και να μπορούν να βρουν ευκαιρίες αγοράς-πώλησης χρηματοοικονομικών μέσων μέσα από ένα αυτοματοποιημένο σύστημα που παρέχει ο Web Server.

Abstract.

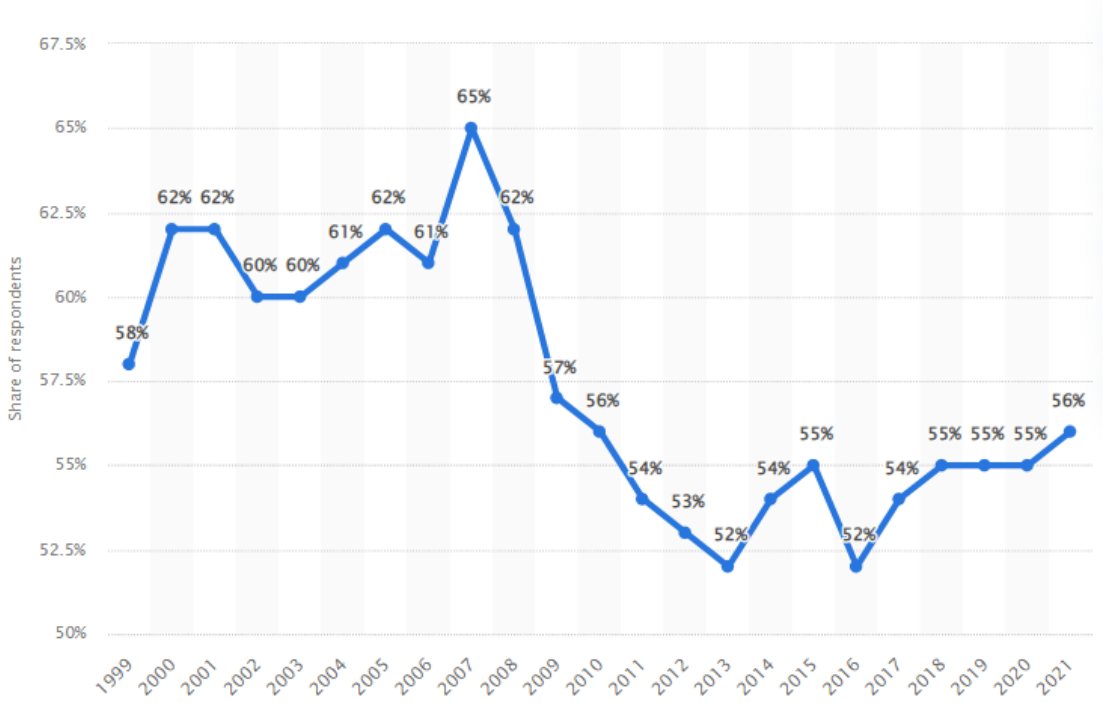
Due to the rapid development of technology, financial assets are a more accessible way of investing money than ever. They serve the flow of capital and with proper market research, they could lead to profits for their investors. The goal of this thesis is the development of a Full Stack Web Application, named Financial Assets Analyst. The technologies that were used for the backend included Spring Boot, MongoDB and for the frontend mostly Typescript (NextJS-React). Through this application users can access financial charts, make use of technical analysis indicators, download in CSV format the data, and find buy-sell opportunities through an automated trading system provided by the Web Server.

1 Εισαγωγή.

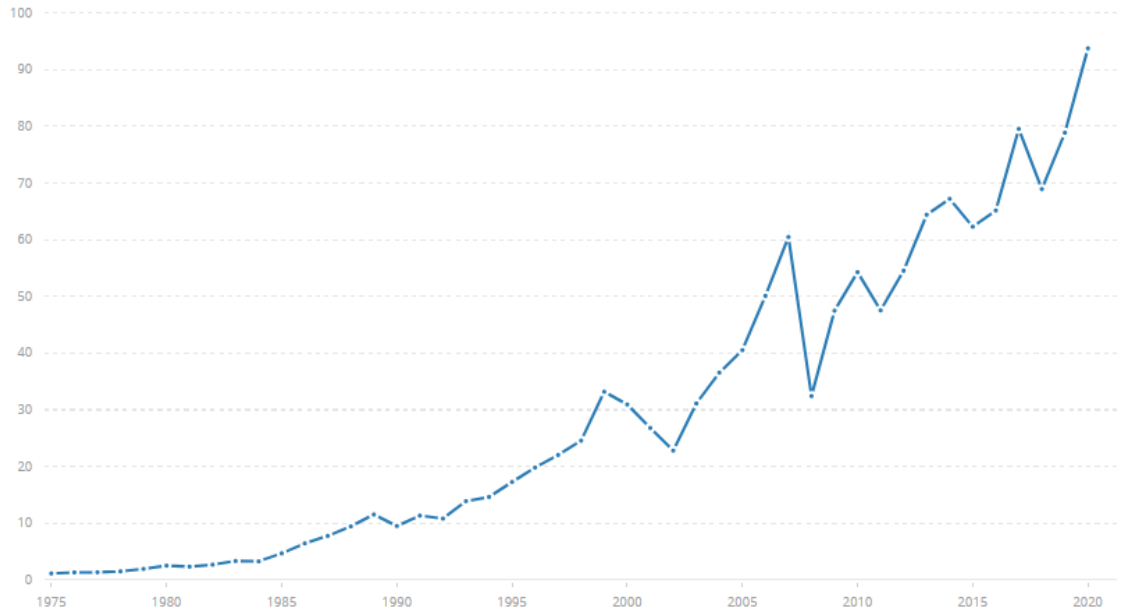
Στόχος μιας επένδυσης είναι να αποφέρει όσο το δυνατόν περισσότερα κέρδη, ανάλογα με το προφίλ κινδύνου. Στις περισσότερες επενδύσεις υπάρχει πάντα ο κίνδυνος να χαθεί μεγάλο μέρος το αρχικού κεφαλαίου. Για να περιοριστεί ο κίνδυνος αυτός έχουν εφευρεθεί πολλές τεχνικές (π.χ. machine learning, trading patterns κ.α.) με τη πιο δημοφιλή να είναι οι δείκτες τεχνικής ανάλυσης. Πολλές φορές η πρόβλεψη της τιμής μιας μετοχής δεν μπορεί να υπολογιστεί με ακρίβεια, και αυτό οφείλεται στο ότι οι αγορές ενσωματώνουν τεράστιο όγκο «παραμέτρων» που καθορίζουν τη τιμή της. Τέτοια αφορούν μετοχικά συμβούλια, η κατάσταση του παγκόσμιου οικονομικού ιστού, ειδήσεις οικονομικού χαρακτήρα αλλά ακόμα και η τυχαιότητα των αγορών.

Πριν από την ύπαρξη του παγκόσμιου ιστού, η διαδικασία της επένδυσης ήταν περίπλοκη, χρονοβόρα και κοστοβόρα. Για να επενδύσει κάποιος στο χρηματιστήριο έπρεπε συνήθως να επικοινωνήσει με το χρηματιστήριο για τις συναλλαγές που θέλει να πραγματοποιήσει. Έπειτα ο χρηματιστής που τον αναλάμβανε έπρεπε να εκτελέσει την εντολή αγοράς. Η όλη αυτή διαδικασία λόγω της διαμεσολάβησης για την εκτέλεση της εντολής οδηγούσε σε «ανακριβές» κλείδωμα της τιμής αγοράς του χρηματοοικονομικού μέσου, διότι η τιμή του μεταβάλλεται συνεχώς. Επίσης για την εκτέλεση της εντολής υπήρχαν μεγάλα ποσοστά προμηθειών τα οποία λάμβαναν οι χρηματιστές.

Στις μέρες μας η αγορά χρηματοοικονομικών μέσων έχει απλουστευθεί και επιταχυνθεί σε μεγάλο βαθμό. Με τη βοήθεια του παγκόσμιου ιστού έχουν δημιουργηθεί πλατφόρμες στις οποίες ο τελικός χρήστης μπορεί να εκτελεί άμεσα εντολές αγοράς – πώλησης ενός χρηματοοικονομικού μέσου από οποιοδήποτε χρηματιστήριο του κόσμου. Η αυτοματοποίηση της διαδικασίας αυτής έχει ως αποτέλεσμα την μείωση των ποσοστών των προμηθειών, και την ενίσχυση της αμεσότητας, εφόσον οι συναλλαγές πραγματοποιούνται με το πάτημα ενός κουμπιού. Από έρευνες που πραγματοποιήθηκαν, φάνηκε πως το 2020 πάνω από το 50% των νοικοκυριών στις Ηνωμένες πολιτείες Αμερικής έχουν κάποια επένδυση σε μετοχές (Parker & Fry, 2020), πράγμα που δηλώνει την χρησιμότητα μιας εφαρμογής για τεχνική ανάλυση σε μετοχές, ομόλογα, δείκτες κ.α.



Εικόνα 1 Ποσοστό των ενηλίκων των Ηνωμένων Πολιτειών που έχει επενδύσει στο χρηματιστήριο.

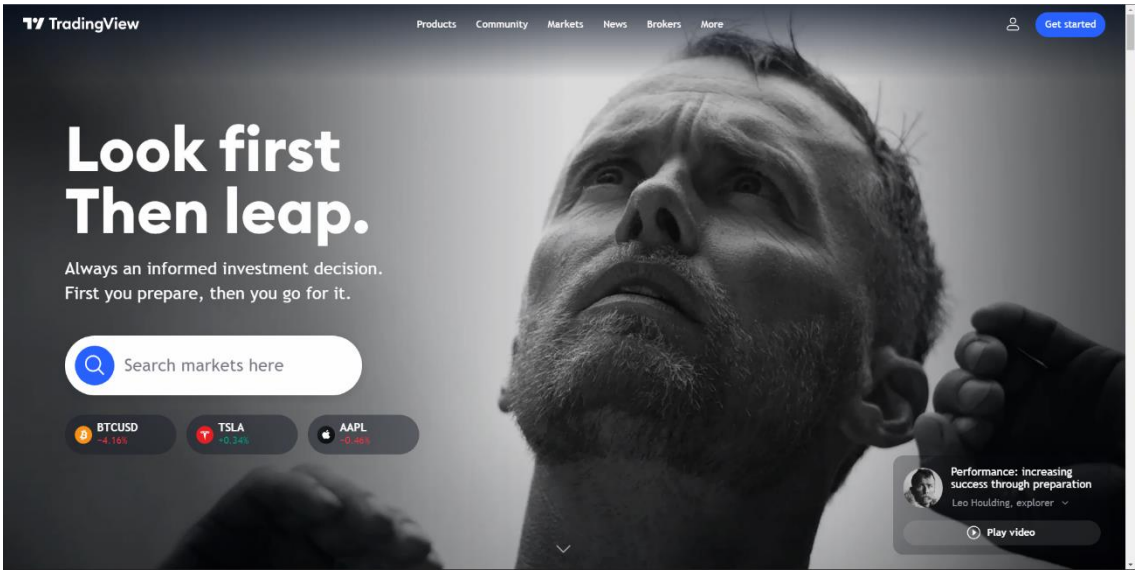


Εικόνα 2: Κεφαλαιοποίηση αγοράς εισηγμένων εγχώριων εταιρειών (σε τρισεκατομμύρια USD).

Στη συγκεκριμένη μεταπτυχιακή διατριβή στοχεύουμε στην υλοποίηση μιας πλατφόρμας η οποία δίνει τη δυνατότητα στους χρηματιστηριακά έμπειρους χρήστες να αξιοποιήσουν δείκτες τεχνικής ανάλυσης και γραφήματα, αλλά και για τους υπόλοιπους να πραγματοποιήσουν λήψη dataset και να συμβουλευτούν ένα αυτοματοποιημένο trading system (συνδυασμό δεικτών τεχνικής ανάλυσης) για την αγορά χρηματοοικονομικών μέσων. Σκοπός της εφαρμογής αυτής είναι να «καλύψει» την όλο και αυξανόμενη ανάγκη των ατόμων να επενδύσουν αποδοτικά στις αγορές.

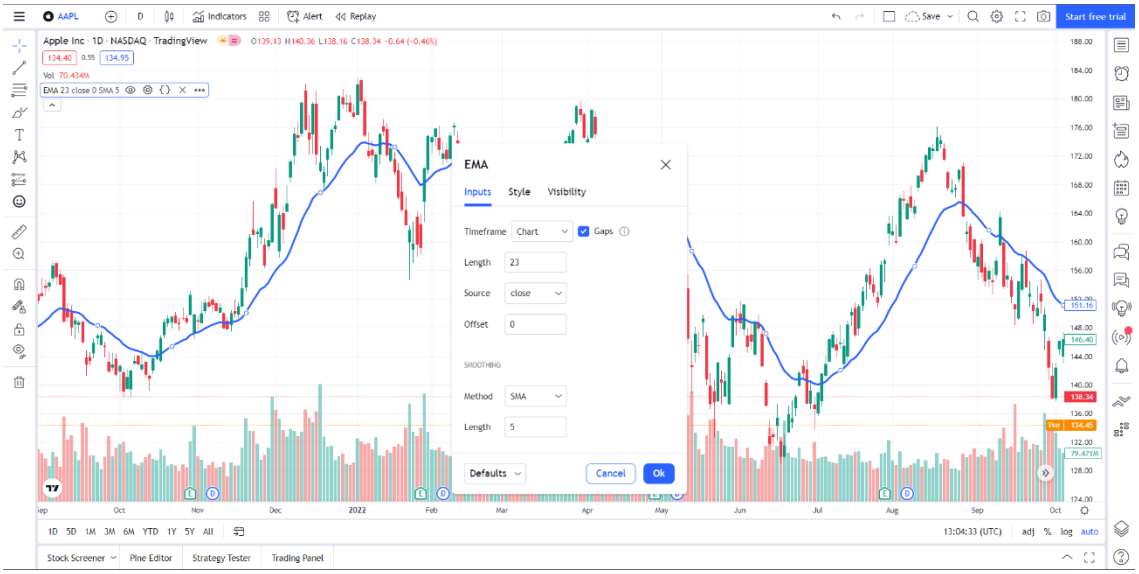
2 Ανασκόπηση Πεδίου.

Η εφαρμογή που θα αναπτυχθεί βασίζεται στη λογική του Tradingview. Το Tradingview ξεκίνησε παρέχοντας γραφήματα και δείκτες τεχνικής ανάλυσης (πχ κινητός μέσος όρος) για να εντοπίσουν οι χρήστες του ευκαιρίες αγοράς-πώλησης. Πλέον έχει επεκταθεί με κοινότητες που συνομιλούν για τα οικονομικά δρώμενα, και δίνει τη δυνατότητα της άμεσης αγοράς-πώλησης χρηματοοικονομικών μέσων μέσα από τη πλατφόρμα του, News feed κ.α.



Εικόνα 3: Tradingview home page.

Εικόνα 4: Tradingview chart Page.



Εικόνα 5: Παραμετροποίηση δεικτών στονTradingview.

Χρησιμοποιώντας την εφαρμογή μπορούμε να κάνουμε κάποιες υποθέσεις για το πως έχει υλοποιηθεί ώστε να ακολουθηθεί παρόμοια στρατηγική στη πτυχιακή εργασία:

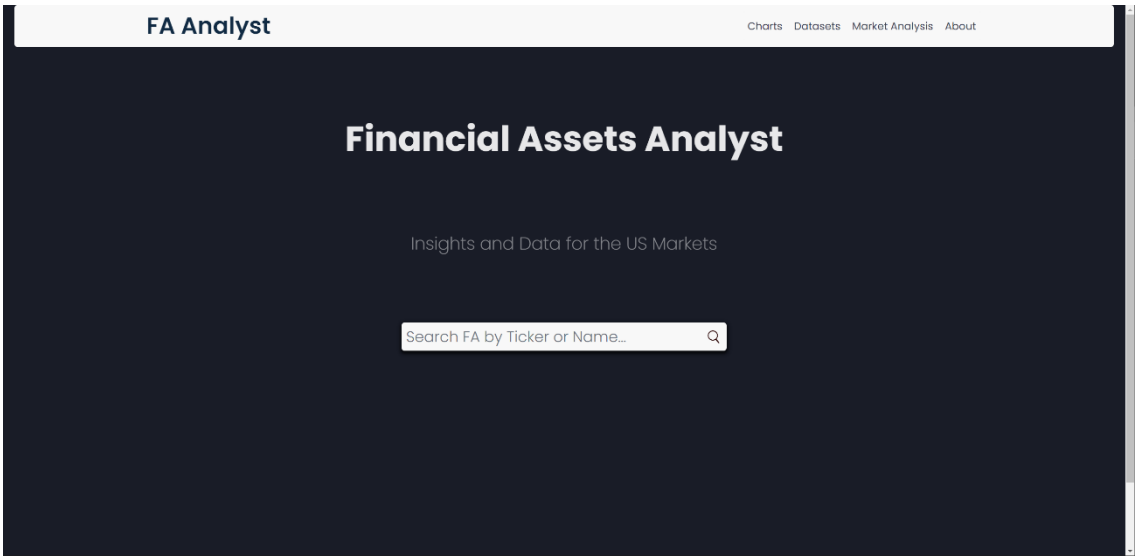
1. Αξιοποιείται κάποιου είδους backend σε μορφή API το οποίο τροφοδοτεί με δεδομένα τα γραφήματα (τιμές Open,High,Low,Close,Volume).
2. Εφόσον οι δείκτες τεχνικής ανάλυσης κατασκευάζονται από τα δεδομένα αυτά, μπορούν να υλοποιούνται είτε στο frontend είτε στο backend. Εάν γίνονται στο backend τότε το υπολογιστικό φορτίο τείνει προς τον server ο οποίος πρέπει να επιστρέψει τα σωστά δεδομένα. Εάν πραγματοποιούνται στο frontend τότε μειώνεται το φόρτο εργασίας του server και οι κλήσεις σε αυτό σε μεγάλο βαθμό αλλά απαιτείται υπολογιστική ισχύς από τον client. Συνήθως ακολουθούμε αρχιτεκτονικές fat server – thin client για τέτοιες εφαρμογές αλλά όχι απαραίτητα.
3. Δίνεται μεγάλη προσοχή στο frontend της εφαρμογής. Ο όγκος δεδομένων είναι πολύ μεγάλος. Εάν απλά δοκιμάζαμε να φορτώσουμε όλα τα δεδομένα σε γράφημα τότε το document object model θα κατέρρεε, δημιουργώντας κακό user experience. Για να επιτυγχάνεται τόσο ομαλή εμπειρία χρήσης πρέπει να χρησιμοποιούνται μέθοδοι virtualization (π.χ. React – virtual DOM).

Επομένως στην εφαρμογή βλέπουμε πως ενσωματώνεται τεράστιος όγκος πληροφοριών και εργαλείων. Σε έναν νέο χρήστη μια τέτοια εμπειρία μπορεί να αποβεί αποθαρρυντική. Για έναν επαγγελματία χρηματιστή θα μπορούσε να είναι απαραίτητα για την εργασία του. Στόχος στη πτυχιακή εργασία είναι να δημιουργηθεί μια όσο το δυνατόν πιο απλουστευμένη εκδοχή του Tradingview που να περιέχει τα απολύτως απαραίτητα για κάθε νέο επενδυτή.

3 Παρουσίαση της Εφαρμογής.

3.1 Γραφήματα Εφαρμογής.

Το πρώτο κομμάτι της εφαρμογής αφορά τα γραφήματα. Σε αυτό οι χρήστες από την αρχική σελίδα παροτρύνονται να αναζητήσουν Αμερικάνικα χρηματοοικονομικά μέσα που τους ενδιαφέρουν.



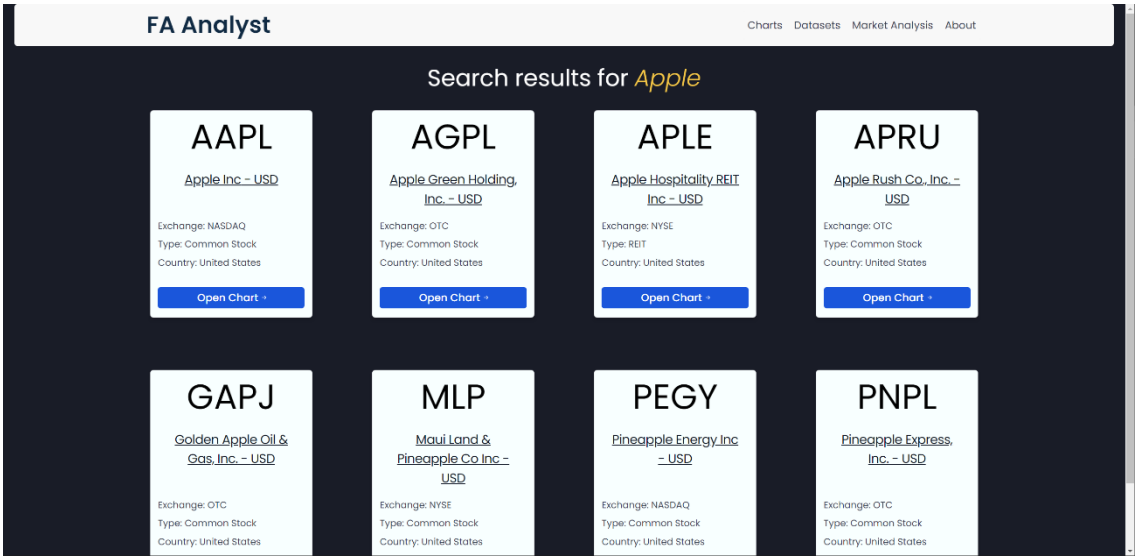
Εικόνα 6: FA Analyst Home page.

Από το πρώτο κιόλας πάτημα στη μηχανή αναζήτησης εμφανίζονται προτεινόμενα χρηματοοικονομικά μέσα με σκοπό τη διευκόλυνση του χρήστη.

Εικόνα 7: FA Analyst, λειτουργία μηχανής αναζήτησης.

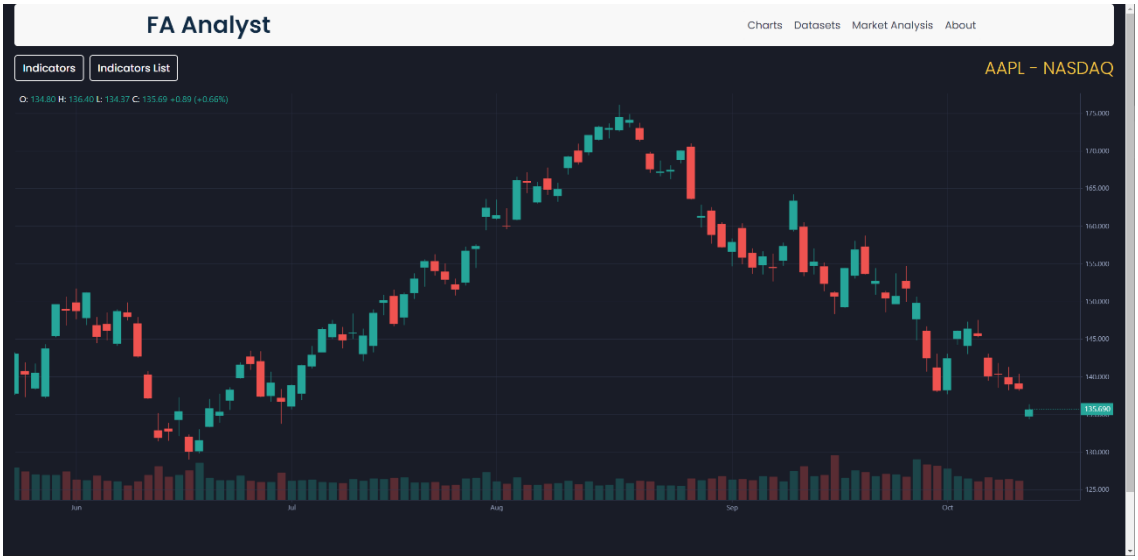
Έπειτα οι χρήστες μεταφέρονται στην σελίδα αναλυτικών αποτελεσμάτων αναζήτησης. Εκεί μπορούν να δουν λίγες παραπάνω πληροφορίες για τα χρηματοοικονομικά μέσα με τη λέξη-κλειδί

που αναζήτησαν και πατώντας κλικ στο κουμπί «Open Chart» να μεταφερθούν στο γράφημα της εφαρμογής.



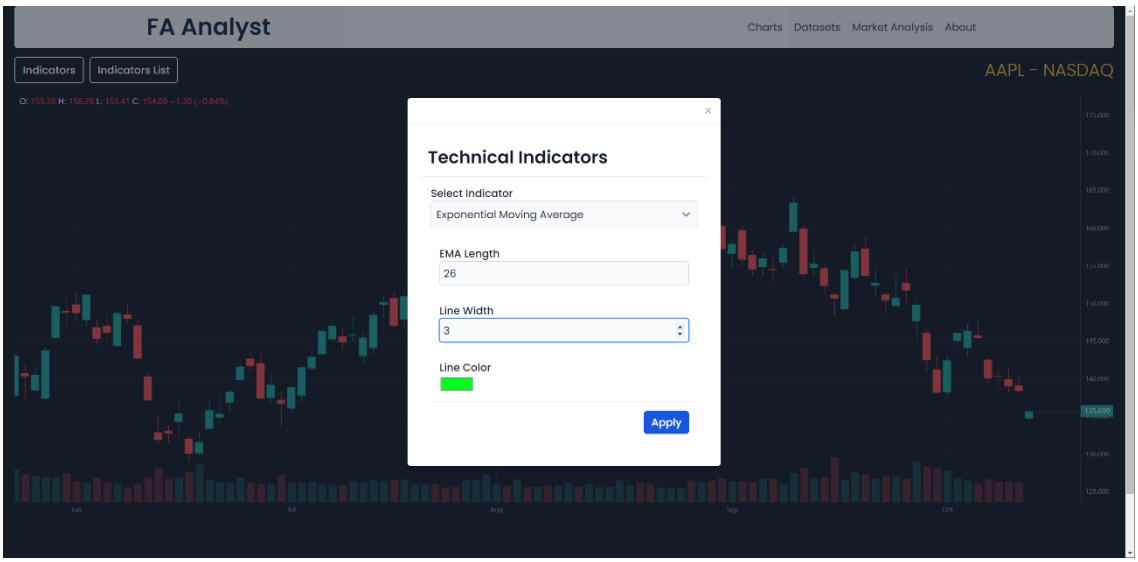
Εικόνα 8:FA Analyst, search results page.

Στη συνέχεια, εμφανίζεται το κατάλληλο γράφημα στον χρήστη σε μορφή Candlestick Chart (Harvey, 2012). Με πράσινο χρώμα απεικονίζεται η ανοδική πορεία της τιμής σε σχέση με τη χθεσινή ενώ με κόκκινο η καθοδική.



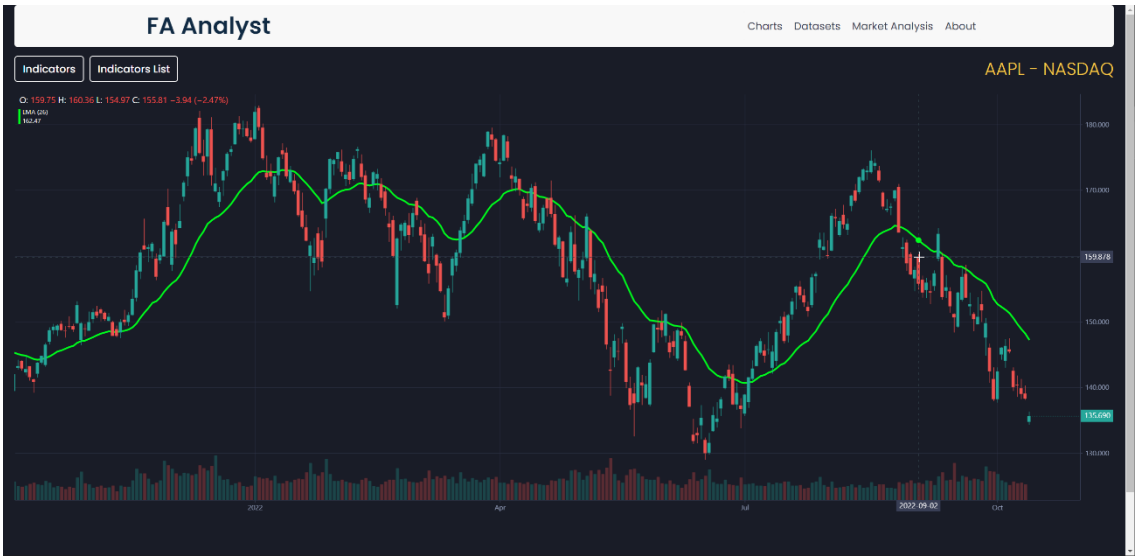
Εικόνα 9: FA Analyst, chart page.

Πατώντας στο κουμπί «indicators» εμφανίζεται ένα μενού επιλογής δεικτών τεχνικής ανάλυσης (παρόμοια με το Tradingview που αναλύθηκε παραπάνω) με δυνατότητα παραμετροποίησης αυτών προτού ενταχθεί στο γράφημα.



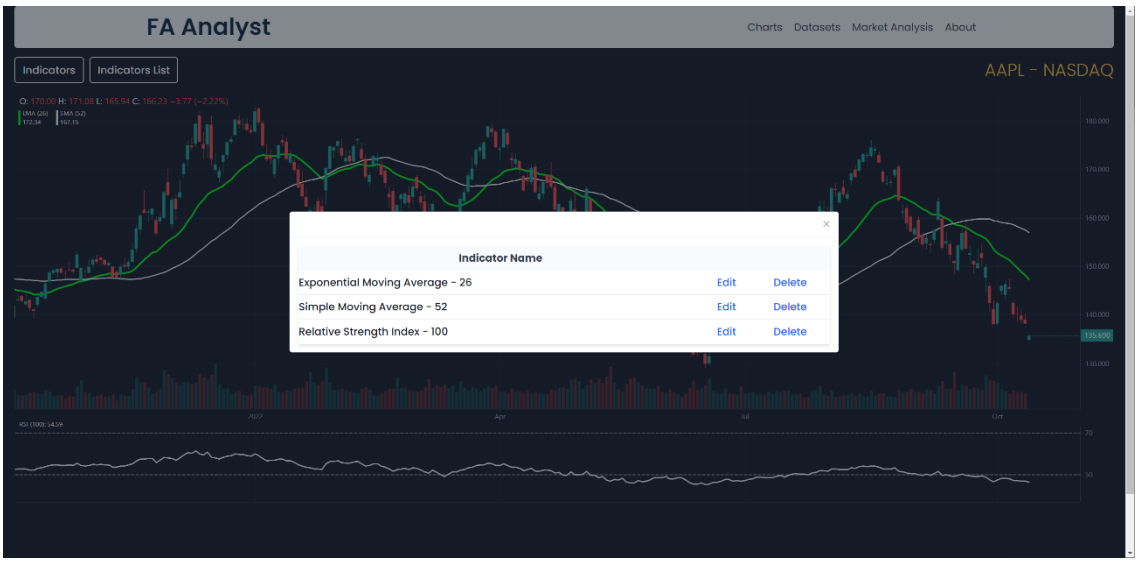
Εικόνα 10: FA Analyst, chart page.

Εφόσον έχουν επιλεχθεί οι επιθυμητοί παράμετροι ο δείκτης θα εμφανιστεί στο γράφημα.



Εικόνα 11: FA Analyst, γράφημα με δείκτη exponential moving average.

Στην εφαρμογή υποστηρίζονται και λειτουργείες update – delete με τη χρήση του κουμπιού «Indicators List». Εμφανίζεται ένα modal με τη λίστα των ενεργών δεικτών τεχνικής ανάλυσης όπου μπορούμε να τους αφαιρέσουμε ή να τους τροποποιήσουμε.



Εικόνα 12: FA Analyst, Indicators List γραφήματος.

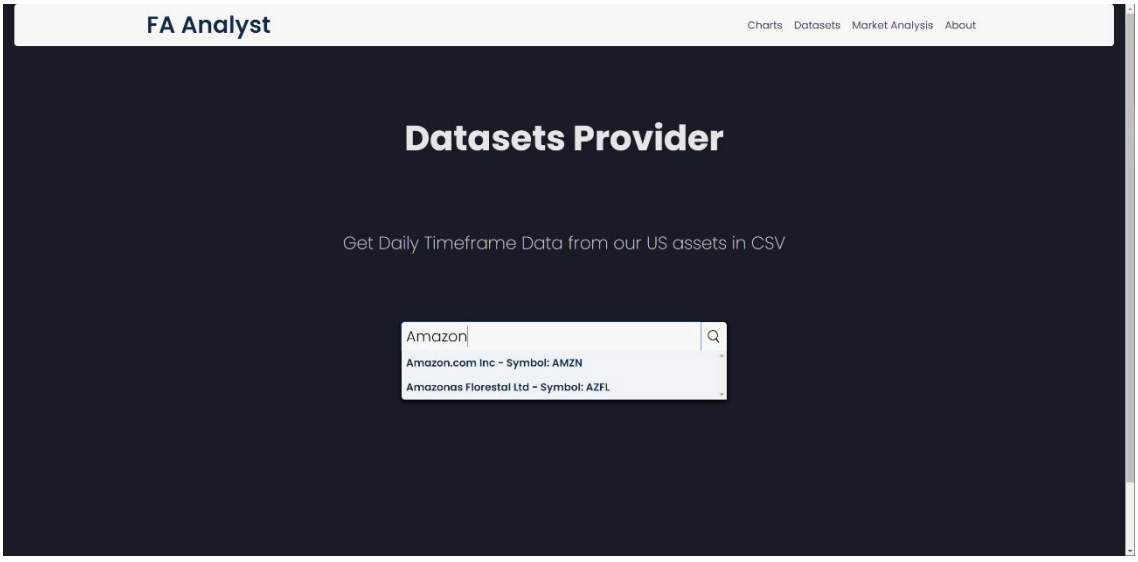
Επομένως όσον αφορά το γράφημα υπάρχουν οι δυνατότητες για CRUD (Create – Read – Update – Delete) δίνοντας την ελευθερία στον χρήστη να προσαρμόσει τα γραφήματα και την στρατηγική επένδυσης ανάλογα με τις ανάγκες του. Προς το παρόν οι δείκτες που υποστηρίζονται είναι :

1) Simple Moving Average (SMA).
2) Exponential Moving Average (EMA).
3) Relative Strength Index (RSI).
4) Moving Average Convergence Divergence (MACD).

3.2 Δεδομένα.

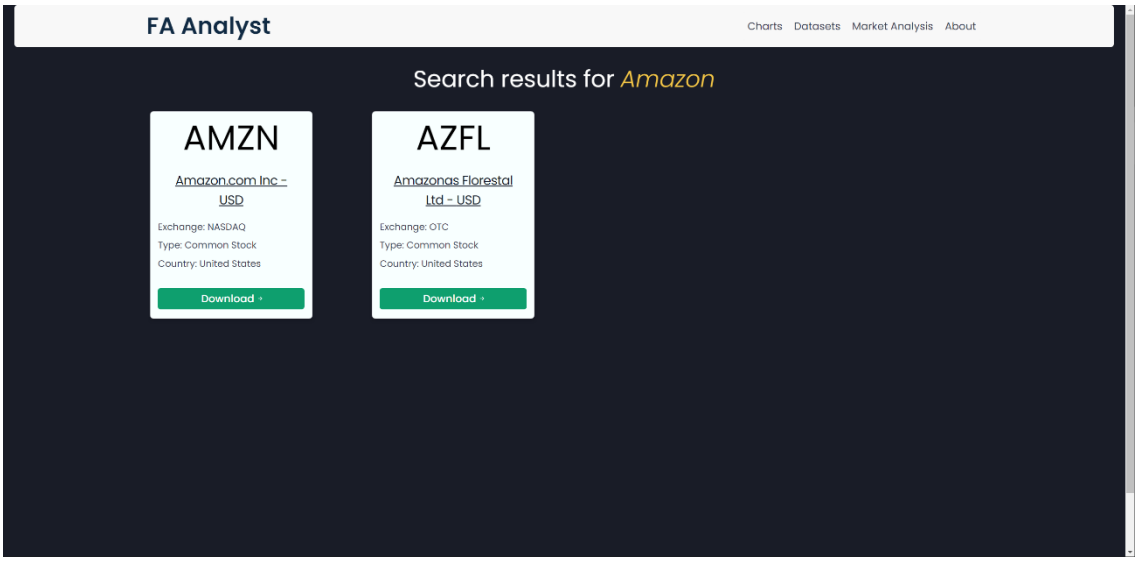
Στο Financial Assets Analyst δίνεται η δυνατότητα να γίνει λήψη των dataset για προσωπική χρήση. Η εξαγωγή γίνεται σε μορφή CSV. Αυτά τα dataset μπορούν να φανούν χρήσιμα σε data engineers για τη πρόβλεψη ή για testing στρατηγικών επένδυσής (backtesting, συνήθως σε γλώσσα προγραμματισμού Python).

Σε παρόμοιο μοτίβο με το κομμάτι των γραφημάτων υπάρχει μια μηχανή αναζήτησης όπου ο χρήστης πληκτρολογεί το χρηματοοικονομικό μέσο που επιθυμεί και παραπέμπεται στη σελίδα αποτελεσμάτων.



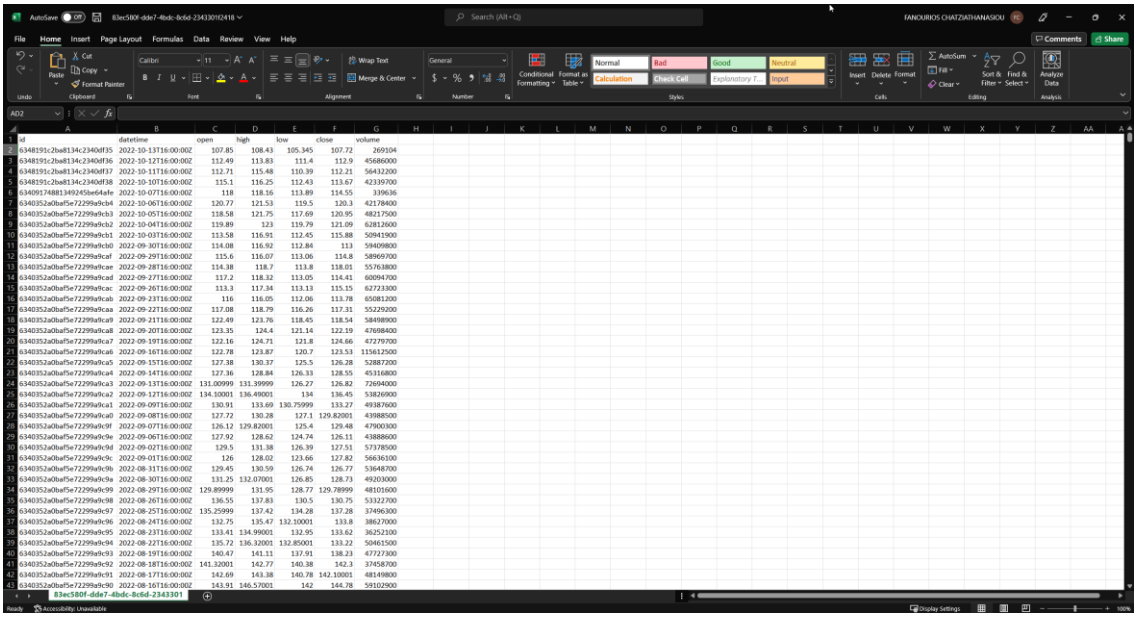
Εικόνα 13: FA Analyst, Datasets home page.

Ακολουθώντας παρόμοιο στυλ για την ομοιομορφία της εφαρμογής, εμφανίζονται λίγες παραπάνω πληροφορίες στα αποτελέσματα αναζήτησης καθώς και η δυνατότητα να πραγματοποιηθεί λήψη του dataset, πατώντας το κουμπί «Download».



Εικόνα 14: FA Analyst, αποτελέσματα αναζήτησης σε Datasets.

Τέλος, μπορούμε να ανοίξουμε το αρχείο και βλέπουμε όλα τα χρηματοοικονομικά δεδομένα που προσφέρει η εφαρμογή για το συγκεκριμένο μέσο , σε ένα CSV αρχείο.

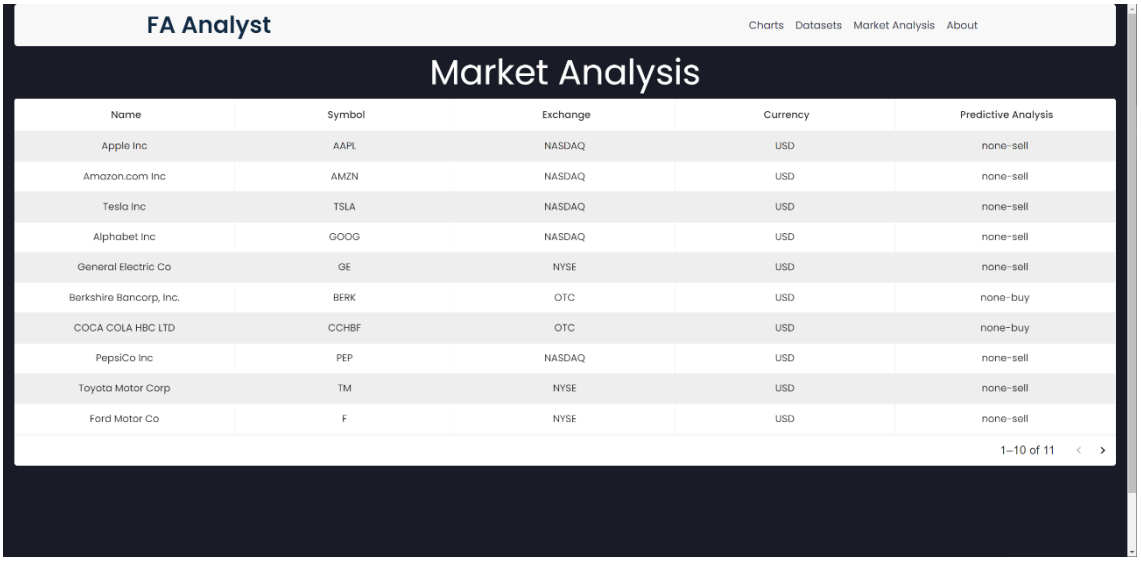


Εικόνα 15: CSV αρχείο από τη λήψη μέσω του Dataset provider του FA Analyst

3.3 Ανάλυση Αγορών.

Τελευταίο μέρος της εφαρμογής είναι η ανάλυση των αγορών. Σε αυτό εμφανίζονται σε μορφή data grid τα δεδομένα που βρίσκονται αποθηκευμένα στη MongoDB βάση δεδομένων. Κάθε γραμμή αποτελείται και από ένα χρηματοοικονομικό μέσο με πληροφορίες για αυτό. Στη τελευταία στήλη αναγράφεται η αυτοματοποιημένη πρόβλεψη που έχει υλοποιηθεί στο backend για την αγορά ή πώληση. Πιο συγκεκριμένα η στήλη Predictive Analysis μπορεί να πάρει τις παρακάτω κατηγορικές τιμές:

1) none-sell: Δεν πληρούνται όλα τα κριτήρια της στρατηγικής αλλά μπορεί να εμφανιστεί προοπτική πώλησης στο σύντομο μέλλον (μπορεί πχ 2/3 δείκτες να σηματοδοτούν πώληση αλλά όχι και ο 3ος).
2) none-buy: Δεν πληρούνται όλα τα κριτήρια της στρατηγικής αλλά μπορεί να εμφανιστεί προοπτική αγοράς στο σύντομο μέλλον (μπορεί πχ 2/3 δείκτες να σηματοδοτούν αγορά αλλά όχι και ο 3ος).
3) Buy: Όλοι οι κανόνες της στρατηγικής πληρούνται και προτείνουν την αγορά του συγκεκριμένου χρηματοοικονομικού μέσου.
4) Buy: Όλοι οι κανόνες της στρατηγικής πληρούνται και προτείνουν την πώληση του συγκεκριμένου χρηματοοικονομικού μέσου.



Εικόνα 16: FA Analyst, Market Analysis home page.

Πατώντας το αριστερό ή το δεξί βέλος κάτω-δεξιά στο data grid μπορούμε να περιηγηθούμε σε όλα τα χρηματοοικονομικά δεδομένα σε μορφή pagination.

4 Αρχιτεκτονική Συστήματος

Ενδιαφέρον έχει να δούμε τα τεχνικά κομμάτια υλοποίησης της εφαρμογής αυτής. Θα μπορούσαμε να τη διαχωρίσουμε σε 2 τμήματα: το frontend (UI- για τον end user) και το backend (Διαχείριση δεδομένων).

4.1 Backend

4.1.1 Εύρεση Δεδομένων

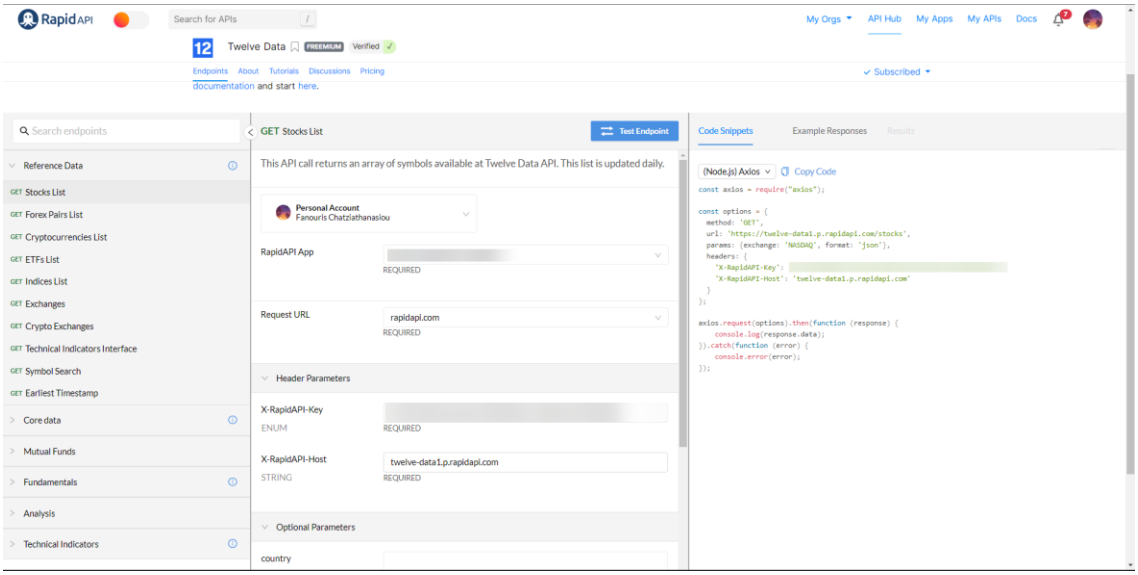
Ξεκινώντας από το backend το πρώτο ερώτημα που πρέπει να απαντηθεί είναι το πού θα βρούμε χρηματιστηριακά δεδομένα. Έπειτα από έρευνα καταλήξαμε σε 2 διαφορετικές εναλλακτικές:

1. **Web scraping:** Μέσω ενός automated browser να αντλούμε τα δεδομένα από κάποια τρίτη ιστοσελίδα (πχ Google Finance) που τα παρέχει.
2. **Third Party API:** Χρησιμοποιώντας κάποιο free tier από Financial API providers να έχουμε πρόσβαση σε δεδομένα.

Παρόλο που το web scraping εκ πρώτης όψεως φαίνεται ως μια επιλογή ιδανική για να έχουμε πρόσβαση σε δεδομένα, περιορίζεται από πολλούς παράγοντες. Εάν αλλάξει η δομή της ιστοσελίδας που διενεργούμε scraping τότε το automation δεν θα λειτουργεί. Επίσης ένα σωστά δομημένο website εταιρείας μεγάλης κεφαλαιοποίησης (πχ Amazon.com) ενσωματώνει συστήματα anti-scraping που αποτρέπουν bots από το να εξάγουν πληροφορία από την ιστοσελίδα τους. Επιπλέον, εάν η ιστοσελίδα κατασκευάζεται στον client, μπορεί να μην έχουν φορτώσει όλα τα δεδομένα τη στιγμή που πραγματοποιείται scraping και να οδηγηθούμε σε ανακρίβειες. Έτσι παρατηρούνται πολλά εμπόδια που μπορούν να εμποδίσουν την ομαλή λειτουργία της εφαρμογής.

Η δεύτερη επιλογή είναι απλά να βρούμε ένα API που παρέχει τα δεδομένα αυτά και μέσω του free plan του να έχουμε πρόσβαση σε δωρεάν δεδομένα. Ο περιορισμός εδώ επικεντρώνεται κυρίως στις κλήσεις του API (rate limits, πχ 50/ ώρα). Ένα τέτοιο API είναι της **Twelve Data** που

παρέχεται και μέσω του RapidApi.com. Αυτή είναι και η εναλλακτική που θα επιλεχθεί για τη συγκεκριμένη εργασία.



Εικόνα 17: Rapidapi.com Twelve data API testing dashboard.

Στην ιστοσελίδα αυτή βλέπουμε πληθώρα δεδομένων που μπορούν να αξιοποιηθούν. Στο free plan περιοριζόμαστε σε χρηματοοικονομικά μέσα των Ηνωμένων Πολιτειών Αμερικής , αλλά εάν θεωρητικά πληρώναμε για το premium plan η εφαρμογή αυτομάτως θα επεκτεινόταν για όλα τα χρηματοοικονομικά μέσα. Επίσης, παρατηρούμε ότι προσφέρονται και δείκτες τεχνικής ανάλυσης από το συγκεκριμένο API endpoint. Στα πλαίσια αυτής της εργασίας όμως θα κατασκευάσουμε τους δείκτες στο backend για να μην διενεργούμε κλήσεις στο freemium API οι οποίες θεωρητικά είναι περιττές αφού έχουμε τα δεδομένα να τους αναπαράγουμε μόνοι μας.

4.1.1 Βάση Δεδομένων – Ανάλυση εναλλακτικών (SQL – NoSQL)

Για την επιλογή της βάσης δεδομένων για το συγκεκριμένο project πρέπει να αναλυθούν οι εναλλακτικές που υπάρχουν. Αυτές είναι οι σχεσιακές βάσεις δεδομένων (SQL) και οι μη σχεσιακές βάσεις δεδομένων (NoSQL).

Οι SQL(Structured Query Language) (Smallcombe, 2021) βάσεις στηρίζονται σε προσχεδιασμένα schemas τα οποία καθορίζουν τη μορφή των δεδομένων. Επομένως γίνεται αντιληπτό πως για τη δημιουργία βάσης SQL πρέπει να τεθούν αυστηροί κανόνες που θα καθορίζουν πως πρέπει να είναι τα δεδομένα, και έτσι όλα τα δεδομένα που εισάγουμε στη βάση πρέπει να ακολουθούν αυτούς τους κανόνες.

Οι NoSQL (Not only SQL) βάσεις δεδομένων έχουν δυναμικά schemas για μη δομημένα δεδομένα, και έτσι μπορούν να αποθηκεύονται σε πολλές μορφές. Συνήθως αποθηκεύονται σε μορφή key-value pair. Αυτή η ευελιξία προσφέρει:

1. Τη δυνατότητα να δημιουργούμε έγγραφα χωρίς να προσχεδιάσουμε τη δομή αυτών.
2. Να έχουμε διαφορετική σύνταξη εντολών των βάσεων από τη μία βάση δεδομένων στην άλλη (πχ Cassandra – MongoDB).
3. Μπορούν να παραμετροποιηθούν-προστεθούν τα πεδία του κάθε έγγραφου μετά την δημιουργία και λειτουργία της βάσης με ευκολία.(πχ σε ένα έγγραφο που αφορά τους χρήστες μιας υπηρεσίας, να προστεθεί το πεδίο του αριθμού του κινητού τους τηλεφώνου).

Σχετικά με την επεκτασιμότητα των δύο αρχιτεκτονικών αυτών, οι SQL βάσεις είναι καθέτως επεκτάσιμες πράγμα που σημαίνει ότι μπορεί να προστεθεί επιπλέον μνήμη ή επεξεργαστική ισχύ

στον υπάρχων server για καλύτερες αποδόσεις. Από την άλλη, οι NoSQL είναι οριζοντίως επεκτάσιμες, δηλαδή μπορούν να τμηματοποιήσουν την μεγάλη κινητικότητα προσθέτοντας επιπλέον NoSQL servers στη συγκεκριμένη βάση. Από τις δύο η NoSQL κυρίως προτιμάται για πολλά και συχνώς μεταβαλλόμενα δεδομένα.

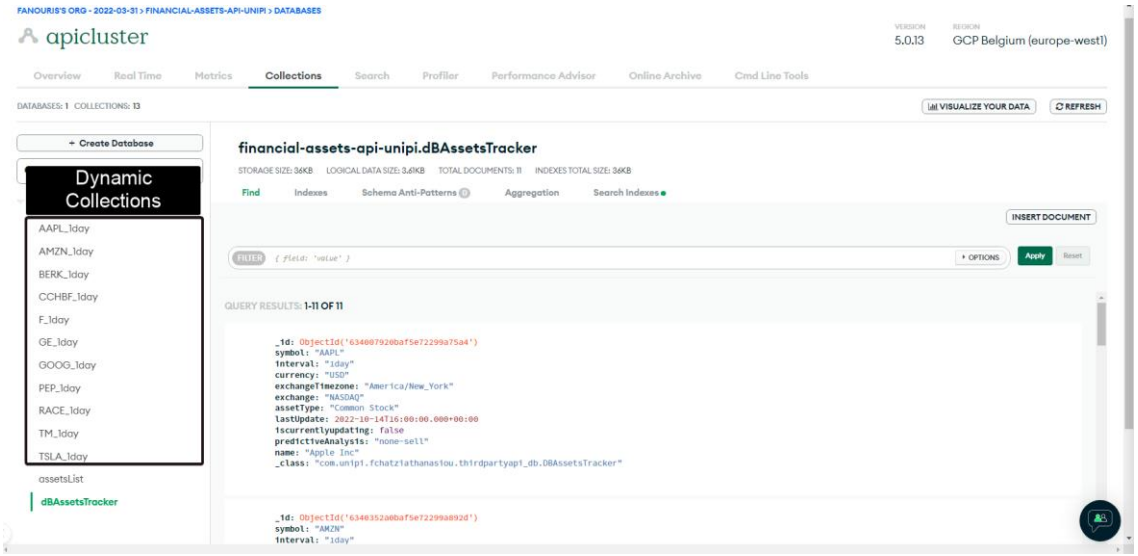
4.1.2 Βάση Δεδομένων – Επιλογή της MongoDB (NoSQL)

Η MongoDB πρωτοεμφανίστηκε το 2009 και αποτελεί μία από τις πιο δημοφιλείς βάσεις δεδομένων. Η αρχιτεκτονική της στηρίζεται στη διαχείριση αρχείων JSON χωρίς τους αυστηρούς περιορισμούς που θέτει μια σχεσιακή βάση δεδομένων(NoSQL που αναλύεται στην προηγούμενη ενότητα). Έτσι προσφέρεται μια μεγαλύτερη ελαστικότητα στη παραμετροποίηση της βάσης σε συνδυασμό με την απλή μορφή της.

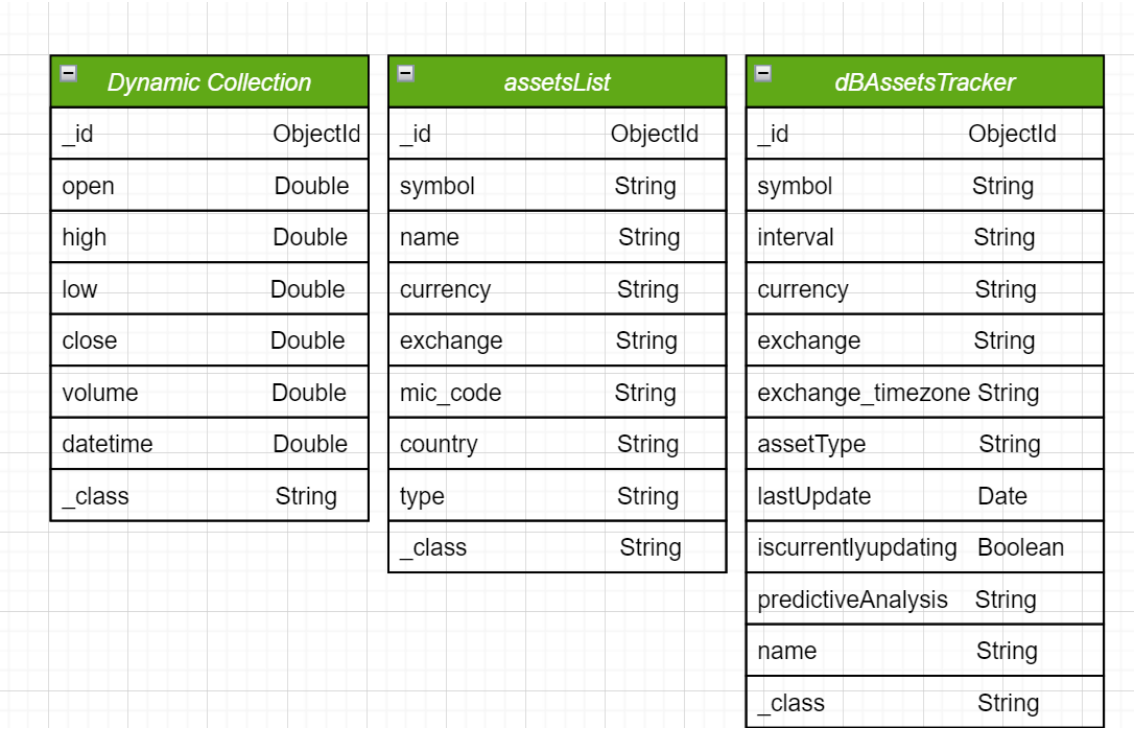
Ένας ακόμη λόγος να προτιμηθεί η MongoDB στο συγκεκριμένο project είναι το γεγονός ότι προμηθευόμαστε τα δεδομένα από έναν 3ο πάροχο, πράγμα το οποίο δεν εγγυάται την ακεραιότητα των δεδομένων. Κενά πεδία ή διαφορετικό format μεταβλητών μπορεί πιο εύκολα να δημιουργήσει προβλήματα στην εισαγωγή δεδομένων σε μια σχεσιακή βάση.

Στη MongoDB οι αντίστοιχοι πίνακες των σχεσιακών βάσεων ονομάζονται collections. Για τη συγκεκριμένη εργασία τα collections κατασκευάζονται δυναμικά στο backend. Οι τύποι των collection είναι οι εξής:

4. Dynamic collections με το όνομα του χρηματοοικονομικού asset ακολουθούμενο από το timeframe (πχ AAPL_1day). Το κάθε ένα από αυτά τα collections περιέχει time series με δεδομένα του συγκεκριμένου asset.
5. assetsList: Περιέχει όλα τα διαθέσιμα χρηματοοικονομικά μέσα που μπορούν να ληφθούν από το rapidapi.com. Επομένως περιέχει κάθε χρηματοοικονομικό μέσο από μία φορά, μαζί με κάποιες βασικές πληροφορίες του.
6. dBAssetsTracker: Παρακολουθεί όλα τα Dynamic collections με metadata, όπως το εάν πραγματοποιείται αυτή τη στιγμή διαδικασία update, πότε πραγματοποιήθηκε το τελευταίο update.



Εικόνα 18: Δομή της MongoDB Atlas Database.

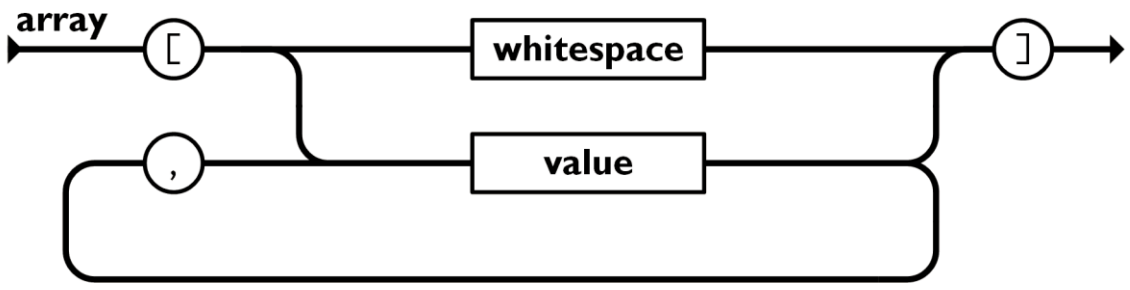


Εικόνα 19: MongoDB Atlas Collections.

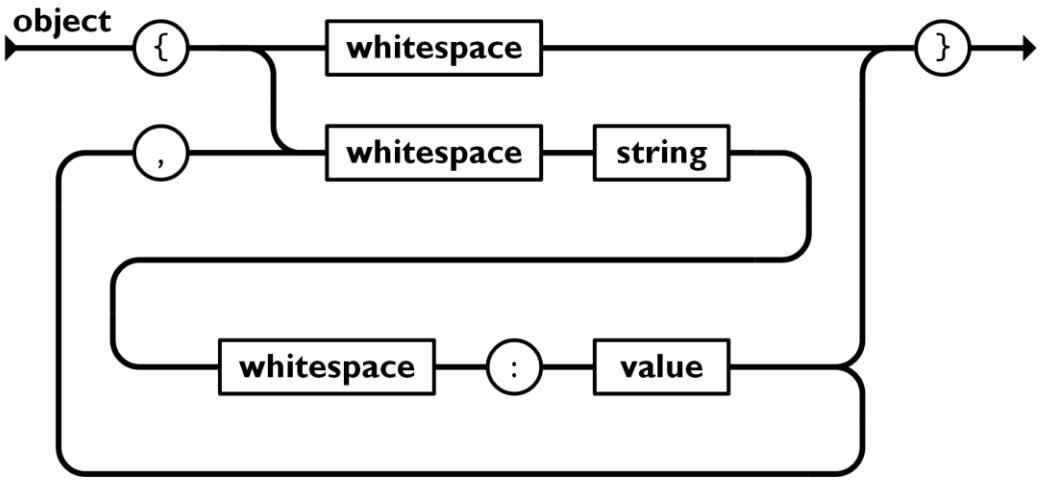
Στη πτυχιακή εργασία η συγκεκριμένη βάση βρίσκεται στο cloud Και για την επικοινωνία με αυτή αρκεί να δώσουμε τα κατάλληλα διαπιστευτήρια στο backend.

4.1.3 Rest API - JSON

Μια από τις πιο διαδεδομένες τεχνικές υλοποίησης για web services είναι η REST (REpresentational State Transfer). Η REST (Gitonga, 2021) (Gupta, 2022) επινοήθηκε αρχικά από τον Roy Fielding ο οποίος εφεύρε και το πρωτόκολλο HTTP. Το REST API είναι μια ενδιάμεση διεπαφή προγραμματισμού εφαρμογών που επιτρέπει σε δύο εφαρμογές να επικοινωνούν μεταξύ τους μέσω HTTP. Τα μηνύματα που ανταλλάσσονται μεταξύ του web server και του client είναι συνήθως σε μορφή JSON. Το JSON (JavaScript Object Notation) είναι μια ελαφριά μορφή ανταλλαγής δεδομένων. Βασίζεται σε ένα υποσύνολο της JavaScript (ECMA-262 3rd Edition - Δεκέμβριος 1999). Το JSON είναι μια μορφή κειμένου που είναι εντελώς ανεξάρτητη από τη γλώσσα, αλλά χρησιμοποιεί συμβάσεις που είναι γνωστές στους προγραμματιστές. Έτσι, η JSON καθίσταται ιδανική γλώσσα ανταλλαγής δεδομένων.



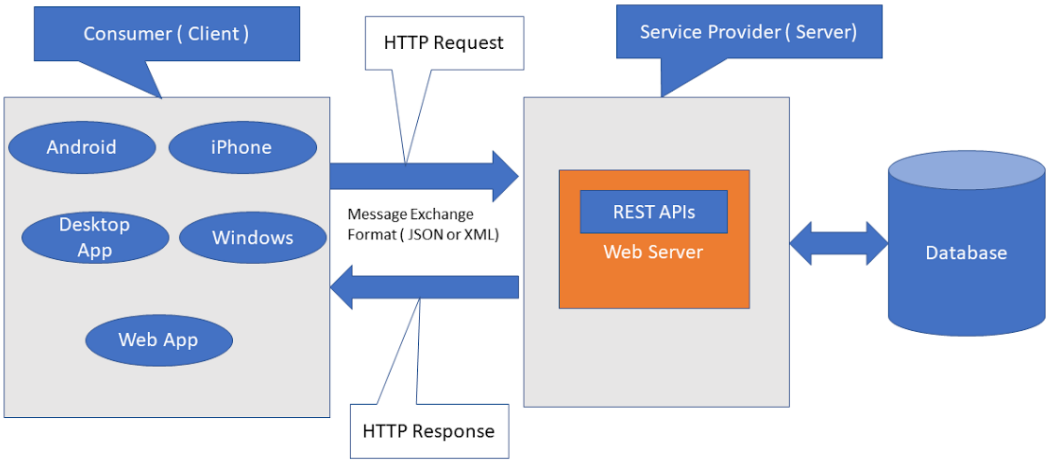
Εικόνα 20: Αναπαράσταση πινάκων με JSON.



Εικόνα 21 : Αναπαράσταση αντικειμένων με JSON.

Συνεπώς, σε μια αρχιτεκτονική REST, ο client και ο server επικοινωνούν μεταξύ τους μέσω μηνυμάτων. Ο client στέλνει ένα μήνυμα στον server (request) και ο server στέλνει την ανάλογη απάντηση στο μήνυμα (response). Ταυτόχρονα μεταφέρονται και μεταδεδομένα ως επικεφαλίδες των μηνυμάτων (request & response headers).

Με την δομή αυτή επιτυγχάνεται πρόσβαση πολλών εφαρμογών σε κοινούς πόρους, πράγμα που προσφέρει προοπτικές επεκτασιμότητας της εφαρμογής σε οποιαδήποτε συσκευή. Για παράδειγμα το ίδιο API μπορεί να δίνει πρόσβαση στα δεδομένα του σε μια web εφαρμογή και σε μια εφαρμογή για κινητές συσκευές.



Εικόνα 22: Αρχιτεκτονική REST σε πλαίσια full stack εφαρμογής.

4.1.4 Web Service - Spring Boot (Java).

Το Spring Framework (Spring, χ.χ.) (Baeldung, χ.χ.) είναι ένα λογισμικό ανοιχτού κώδικα, οι λειτουργίες του οποίου μπορούν να χρησιμοποιηθούν για την ανάπτυξη JAVA εφαρμογών. Αν και το Framework αυτό δεν επιβάλλει κάποιο συγκεκριμένο μοντέλο προγραμματισμού, έχει γίνει δημοφιλές στην κοινότητα της JAVA ως μια προσθήκη στο μοντέλο Enterprise JavaBeans (EJB).

Η πρώτη έκδοση γράφτηκε από τον Rod Johnson, ο οποίος κυκλοφόρησε το Spring Framework με τη δημοσίευση του βιβλίου του “Expert One-On-One J2E Design and Development” τον Οκτώβριο του 2002. Το Spring Framework αρχικά κυκλοφόρησε με την άδεια

Apache 2.0, τον Ιούνιο του 2003. Η πρώτη κυκλοφορία στην παραγωγή, 1.0, έλαβε χώρα το Μάρτιο του 2004. Το Spring 1.2.6 Framework κέρδισε το βραβείο παραγωγικότητας Jolt και το βραβείο JAX Innovation Award το 2006. Το Spring 2.0 κυκλοφόρησε τον Οκτώβριο του 2006, το Spring 2.5 το Νοέμβριο του 2007, το Spring 3.0 το Δεκέμβριο του 2009, το Spring 3.1 το Δεκέμβριο του 2011, και το Spring 3.2.5 το Νοέμβριο του 2013. Το Spring Framework 4.0 κυκλοφόρησε το Δεκέμβριο του 2013. Οι αξιοσημείωτες βελτιώσεις στο Spring 4.0 περιελάμβαναν την υποστήριξη για Java SE (Standard Edition) 8, Groovy 2, ορισμένες πτυχές της Java EE 7, και για WebSocket.

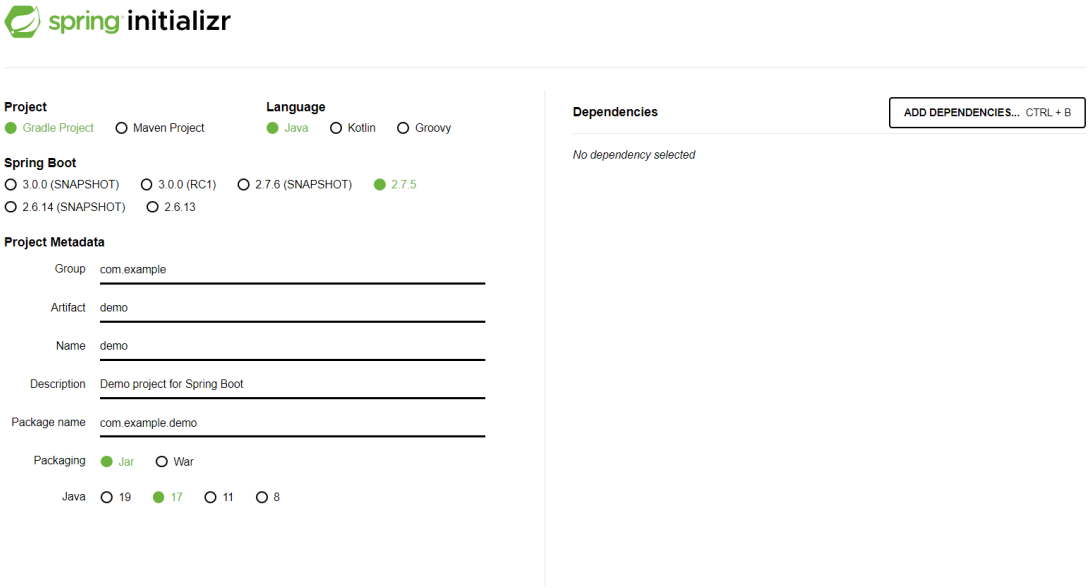
Το Spring Boot 1.0 κυκλοφόρησε τον Απρίλιο του 2014. Το Spring Framework 4.2.0 κυκλοφόρησε στις 31 Ιουλίου 2015 και αναβαθμίστηκε αμέσως στην έκδοση 4.2.1, η οποία κυκλοφόρησε την 1η Σεπτεμβρίου 2015. Το Spring Framework 4.3 κυκλοφόρησε στις 10 Ιουνίου 2016 και θα υποστηρίζεται μέχρι το 2020. Το Spring 5 έχει ανακοινωθεί ότι θα κατασκευαστεί πάνω σε Reactive Streams και θα είναι συμβατό με Reactor Core. Το Spring Framework περιλαμβάνει διάφορες ενότητες/μέρη (modules) που παρέχουν μια σειρά από υπηρεσίες:

- Spring Core Container: είναι το βασικό module του Spring και παρέχει spring containers (Bean Factory και ApplicationContext).
- Aspect-oriented programming: επιτρέπει την υλοποίηση “cross-cutting concerns”.
- Authentication and authorization: διαμορφωμένες διαδικασίες ασφαλείας που υποστηρίζουν μια σειρά προτύπων, πρωτοκόλλων, εργαλείων και πρακτικών μέσω του Spring Security sub-project (formerly Acegi Security System for Spring). Μεταπτυχιακή Διατριβή Αυγέρης Χαράλαμπος Android εφαρμογή για την εύρεση των κοντινότερων συσκευών με χρήση Bluetooth και Spring Boot backend 22.
- Convention over configuration: μείωση του αριθμού των αποφάσεων που χρειάζεται να πάρει ένας προγραμματιστής που χρησιμοποιεί το framework χωρίς όμως να χάνεται η λειτουργικότητα και η ευελιξία.
- Data access: δυνατότητα για εργασία με συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων στην πλατφόρμα Java χρησιμοποιώντας Java Database Connectivity (JDBC) και εργαλεία object-relational mapping καθώς και δυνατότητα για εργασία με NoSQL βάσεις δεδομένων.
- Inversion of control container: παραμετροποίηση των components της εφαρμογής και διαχείριση του κύκλου ζωής των Java objects μέσω του dependency injection.
- Messaging: χρήση Java Message Service (JMS) και βελτίωση της αποστολής μηνυμάτων μέσω standard JMS APIs.
- Model-view-controller: ένα HTTP-and-servlet-based framework που παρέχει τη βάση για web applications και RESTful (representational state transfer Web services).
- Testing: υποστήριξη κλάσεων για εκτέλεση unit tests και integration tests.

Το Java Spring Boot (Spring Boot) είναι μια επέκταση του Spring Framework που κάνει την ανάπτυξη εφαρμογών και microservices ταχύτερη και ευκολότερη. Ουσιαστικά αποτελεί τη λύση του Convention over configuration που αναφέρθηκε παραπάνω, για τη δημιουργία αυτόνομων εφαρμογών, έτοιμων για την παραγωγή, τις οποίες μπορούμε απλώς να εκτελέσουμε. Τα κυριότερα χαρακτηριστικά του είναι:

- Επιτρέπει τη δημιουργία αυτόνομων εφαρμογών (Create stand-alone Spring applications)
- Έχει ενσωματωμένο web server (Embed Tomcat, Jetty or Undertow directly (no need to deploy WAR files)).
- Αυτόματη παραμετροποιήση (autoconfiguration).
- Παροχή λειτουργιών έτοιμων για την παραγωγή (Provide production-ready features).

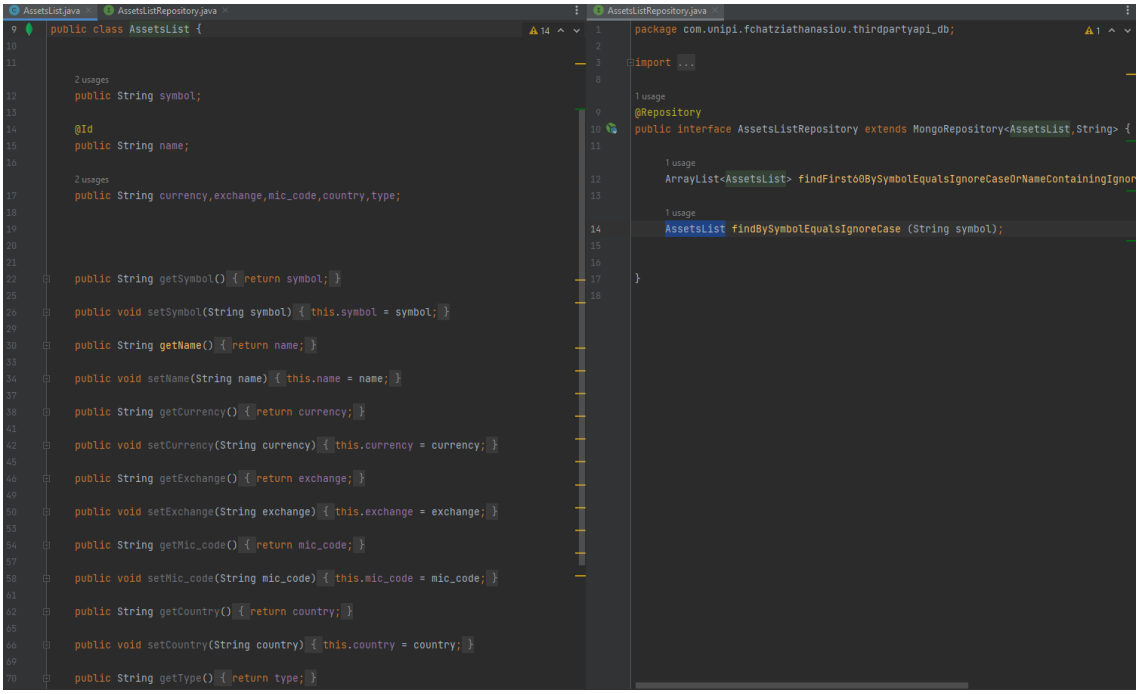
- Δεν υπάρχει απαίτηση για παραμετροποιήση XML (No requirement for XML configuration) Το Spring Boot, λοιπόν, εξαλείφει την ανάγκη των πολύπλοκων ρυθμίσεων που απαιτούνται για την ανάπτυξη μιας Spring εφαρμογής, δίνοντας τη δυνατότητα στον προγραμματιστή να ασχοληθεί με την παραμετροποίηση μόνο συγκεκριμένων κομματιών ανάλογα με τις ανάγκες του project που θέλει να αναπτύξει.



Εικόνα 23: Spring Initializr.

Επομένως, το Spring Framework δίνει τη δυνατότητα να κατασκευάσουμε έναν web server με API endpoints τα οποία θα «χτυπάει» ο end user μέσω του frontend για να λαμβάνει τα ανάλογα δεδομένα.

Τη βάση δεδομένων τη διαχειριζόμαστε μέσω της Java και συγκεκριμένα μέσω ORM (O'Neil, 2008) .Συνοπτικά, ORM είναι η τεχνική της μετατροπής δεδομένων με τη χρήση του αντικειμενοστραφούς προγραμματισμού, δηλαδή απεικονίζουμε και διαχειριζόμαστε τη βάση δεδομένων χωρίς απαραίτητα να συντάξουμε εντολές SQL. Στο Spring το μόνο που έχουμε να κάνουμε είναι να κατασκευάσουμε τις κλάσεις που απεικονίζουν τα collections (model) και τα αντίστοιχα Repositories του, στα οποία εισάγονται όλες οι εντολές που επιθυμούμε για τη συγκεκριμένη κλάση.



```java
// AssetsList.java
public class AssetsList {

    2 usages
    public String symbol;

    @Id
    public String name;

    2 usages
    public String currency,exchange,mic_code,country,type;

    public String getSymbol() { return symbol; }

    public void setSymbol(String symbol) { this.symbol = symbol; }

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }

    public String getCurrency() { return currency; }

    public void setCurrency(String currency) { this.currency = currency; }

    public String getExchange() { return exchange; }

    public void setExchange(String exchange) { this.exchange = exchange; }

    public String getMic_code() { return mic_code; }

    public void setMic_code(String mic_code) { this.mic_code = mic_code; }

    public String getCountry() { return country; }

    public void setCountry(String country) { this.country = country; }

    public String getType() { return type; }
```

```java
// AssetsListRepository.java
package com.unipi.fchatziathanasiou.thirdpartyapi_db;

import ...

1 usage
@Repository
public interface AssetsListRepository extends MongoRepository<AssetsList,String> {

    1 usage
    ArrayList<AssetsList> findFirstóOBySymbolEqualsIgnoreCaseOrNameContainingIgnor

    1 usage
    AssetsList findBySymbolEqualsIgnoreCase (String symbol);


}
```

Εικόνα 24: παράδειγμα κλάσης (αντιστοιχεί σε collection) και του repository του.

Για να διατηρείται το κομμάτι του Market Analysis ενήμερο, πρέπει να οργανωθεί ένα Scheduled task που θα «τρέχε» ανά συγκεκριμένα χρονικά διαστήματα για να εξετάζει εάν υπάρχει προοπτική αγοράς-πώλησης. Ενδεικτικά αυτό το περιθώριο το ορίζουμε στα 15 λεπτά και η υλοποίηση του είναι η εξής:

```java
Fanourios-Chatziathanasiou *
@Scheduled(fixedRate = 900000) //<- every 15 minutes.
public void updateDbAssetsTrackerPredictiveAnalysis () {
    List<DBAssetsTracker> allDbAssets =  dbAssetsTrackerRepository.findAll();

    for (DBAssetsTracker dbAssetsTracker: allDbAssets){
        try {

            dbAssetsTracker.setPredictiveAnalysis(generatePredictiveAnalysis(dbAssetsTracker.symbol,dbAssetsTracker.interval));
        } catch (JSONException e) {

            throw new RuntimeException(e);
        }
    }
    dbAssetsTrackerRepository.saveAll(allDbAssets);
    log.info("{} Predictive Analysis Updated", dateFormat.format(new Date()));
}
```

Εικόνα 25: Scheduled task για την ενημέρωση του Market Analysis.

Παρόμοια πρέπει να διενεργήσουμε και για τη Λίστα με τα χρηματοοικονομικά μέσα που μπορεί η εφαρμογή να παρέχει. Διατηρώντας αυτή τη λίστα, δεν χρειάζεται να αποθηκεύουμε στο backend μας όλα τα δεδομένα των χρηματιστήριών αλλά μόνο όσα έχουν ζητήσει οι χρήστες να δουν έστω και μία φορά.

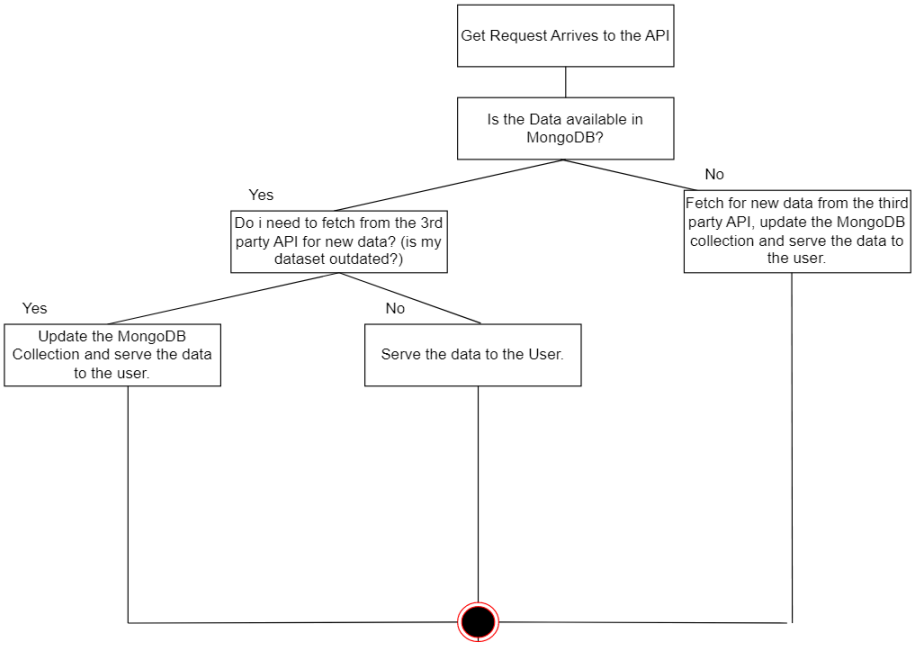
```java
Fanourios-Chatziathanasiou *
@Scheduled(fixedRate = 86400000) // <- once a day
public void updateFinancialAssetsList() throws JSONException {
    JSONObject obj = new JSONObject((String) getAssetsListFromRapidApi());
    JSONArray assetValuesArr = obj.getJSONArray( key: "data");
    CopyOnWriteArrayList<AssetsList> alist = new CopyOnWriteArrayList<AssetsList>();
    for (int i = assetValuesArr.length() - 1; i >= 0; i--) {
        //Create a financialAsset Object and fill the data in order to store it into MongoDB.(will be used in the for loop below),
        //updating the datetime here provides us with the datetime we need to store to the dBAssetsTracker , at the end of the for loop.
        //Which is the last Close Value.
        alist.add(saveAssetsListToDatabase(assetValuesArr.getJSONObject(i)));
    }
    int size = alist.size();
    ArrayList<AssetsList> sharedNames = (ArrayList<AssetsList>) assetsListRepository.findAll();
    for (AssetsList aslst : sharedNames){
        for (AssetsList alst : alist){
            if (aslst.name.equals(alst.getName())){
                alist.remove(alst);
            }
        }
    }
    assetsListRepository.saveAll(alist);
    log.info("{} Available Assets Updated", dateFormat.format(new Date()));

}
```

Εικόνα 256: Scheduled task για τη λίστα χρηματοοικονομικών μέσων.

4.1.5 Αντιμετώπιση περιορισμών δεδομένων εξωτερικού API.

Για την καταπολέμηση του περιορισμού των κλήσεων στο API μπορούμε να ενσωματώσουμε μια βάση δεδομένων σε ρόλο «διαμεσολαβητή». Κάθε φορά που ο χρήστης ζητάει δεδομένα ενός χρηματοοικονομικού μέσου ελέγχεται εάν το συγκεκριμένο μέσο βρίσκεται στη βάση δεδομένων. Ανάλογα με το εάν υπάρχει στη MongoDB βάση και αν είναι ενημερωμένες οι πληροφορίες τότε στέλνουμε τη πληροφορία στο frontend απευθείας ή διενεργούμε κλήση στο εξωτερικό API (Twelve Data). Με αυτόν τον τρόπο αποτρέπεται η επαναλαμβανόμενη κλήση στο Twelve Data για πληροφορίες οι οποίες έχουν επαναληφθεί.



Εικόνα 27: Απλοποιημένη λογική του backend σε διάγραμμα ροής (flowchart).

4.2 Frontend

Όσον αφορά το frontend, υπάρχουν πολλές εναλλακτικές υλοποίησης του. Επιλέχθηκε ως γλώσσα προγραμματισμού η Typescript, μια strictly typed γλώσσα η οποία «απαιτεί» τη δήλωση των μεταβλητών (σε αντίθεση με τη JavaScript). Με αυτόν τον τρόπο επιτυγχάνεται καλύτερη ποιότητα κώδικά ως προς την ανάγνωση, αποσφαλμάτωση και επεκτασιμότητα.

4.2.1 Προκλήσεις του frontend και τρόποι αντιμετώπισης.

Είναι σημαντικό, στο frontend για μια εφαρμογή που διαχειρίζεται μεγάλο όγκο δεδομένων να υπάρχει υψηλή απόδοση στο frontend. Ταυτόχρονα, με το γεγονός ότι το μεγαλύτερο ποσό των χρηστών του παγκοσμίου ιστού χρησιμοποιεί κινητές συσκευές, απαιτείται η δημιουργία μιας ιστοσελίδας με βάση τις αρχές του responsive web design (Nielsen Norman Group, 2014).

Τέλος, σύμφωνα με έρευνες της Google (Google, 2016), 53% των επισκεπτών που περιμένουν πάνω από 3 δευτερόλεπτα για να «φορτώσει» μια ιστοσελίδα, αποχωρεί από αυτή. Επομένως είναι απαραίτητο να αξιοποιηθούν και σύγχρονες τεχνολογίες με γνώμονα το performance του user interface για να δημιουργηθεί μια πιο ελκυστική εφαρμογή προς τους χρήστες της.

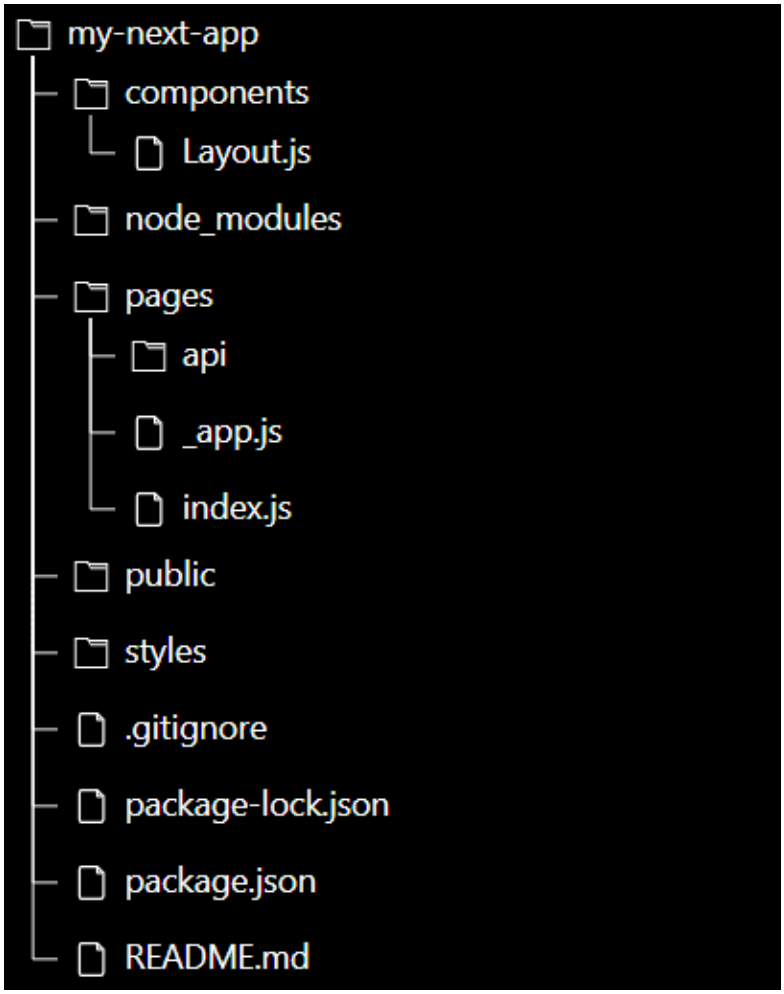
4.2.2 Επιλογή Framework – React & NextJS.

Μαζί με την Typescript αξιοποιούμε την React, τη πιο δημοφιλή open source βιβλιοθήκη για web applications η οποία προσφέρει τη δυνατότητα υψηλού performance στην εφαρμογή. Για την ταχεία ανάπτυξη της εφαρμογής ενσωματώθηκε και το NextJS, ένα framework στη React για το οποίο θα αναφερθούμε αναλυτικά παρακάτω:

Πιο αναλυτικά, η React (Meta Platforms, Inc, χ.χ.) δημιουργήθηκε από τον Jordan Walke, προγραμματιστή της Facebook. Πρωτοεμφανίστηκε με το όνομα FaxJS και ήταν εμπνευσμένη από την XHP, μια βιβλιοθήκη της PHP. Η πρώτη εφαρμογή της React έγινε στην ιστοσελίδα του Facebook το 2011 και ύστερα στο Instagram το 2012. Το πιο επαναστατικό κομμάτι της βιβλιοθήκης αυτής ήρθε στις 18 Απριλίου του 2017, όπου η Facebook ανακοίνωσε το React Fiber, έναν νέο αλγόριθμο για την απόδοση τμημάτων κώδικα στην οθόνη (rendering). Έως τότε, χρησιμοποιούνταν το Stack, το οποίο βασιζόταν στην απόδοση στατικών σελίδων στους χρήστες. Ήταν σχετικά αργό σε περίπλοκες διεργασίες διότι επιχειρούσε να τις επιλύσει ως ένα μεμονωμένο κομμάτι. Από την άλλη, το Fiber στηρίζεται στη τμηματοποίησης των ιστοσελίδων και την αυτόνομή συντήρηση και ενημέρωση τους.

Τα πιο δημοφιλή στοιχεία – κλειδιά που διαφημίζει η React στο documentation της είναι:

- Δηλωτική (Declarative) : Η React κάνει εύκολη τη δημιουργία διαδραστικών UIs. Με αυτή, σχεδιάζονται απλά views για κάθε της εφαρμογής, και η React θα ενημερώσει και θα παρουσιάσει τα σωστά components όταν τα δεδομένα αλλάξουν.Τα declerative views κάνουν τον κώδικά πιο προβλέψιμο και διευκολύνει τον εντοπισμό λαθών.
- Βασισμένη σε τμήματα (Component-Based) : Δίνει τη δυνατότητα δημιουργίας εμφωλευμένων component που διαχειρίζονται το δικό τους state. Η λογική των component είναι γραμμένη σε JavaScript και μπορεί εύκολα να τροποποιηθεί και να λειτουργεί ανεξάρτητα από το Document Object Model των browser.
- Μάθετε μια φορά, γράψτε παντου (Learn Once, Write Anywhere) : Η react είναι code agnostic, δηλαδή λειτουργεί αυτόνομα και ανεξάρτητα από τις τεχνολογίες του υπόλοιπου stack της εφαρμογής.



Εικόνα 28: Παράδειγμα Δομής μιας εφαρμογής NextJS.

Όπως αναφέρθηκε και παραπάνω, μαζί με τη react θα αξιοποιηθεί το NextJS (NextJS, n.d.). Είναι ένα framework κατασκευασμένο πάνω στη React το οποίο δίνει τη δυνατότητα ταχείας ανάπτυξης web εφαρμογών. Παρέχει δομικά στοιχεία που σχετίζονται με:

- User Interface: Πως οι χρήστες «καταναλώνουν» και αλληλοεπιδρούν με την εφαρμογή.
- Routing: Πως οι χρήστες περιηγούνται μεταξύ τμημάτων της εφαρμογής.
- Data Fetching: Πως και πότε γίνεται η λήψη δεδομένων από τον server.
- Rendering: Πως και που να προβάλλεται στατικό ή δυναμικό περιεχόμενο στην ιστοσελίδα.
- Integrations: Πως να επικοινωνήσει η εφαρμογή με υπηρεσίες τρίτων (Content Management Systems, authorization, payments).
- Infrastructure: Πως να κάνουμε deploy τον κώδικα μας.
- Performance: Πως να βελτιστοποιηθεί η απόδοση για τους χρήστες της εφαρμογής.
- Scalability: Η δυνατότητα της προσαρμογής της εφαρμογής καθώς οι προγραμματιστές, τα δεδομένα και η κινητικότητα αυξάνονται.

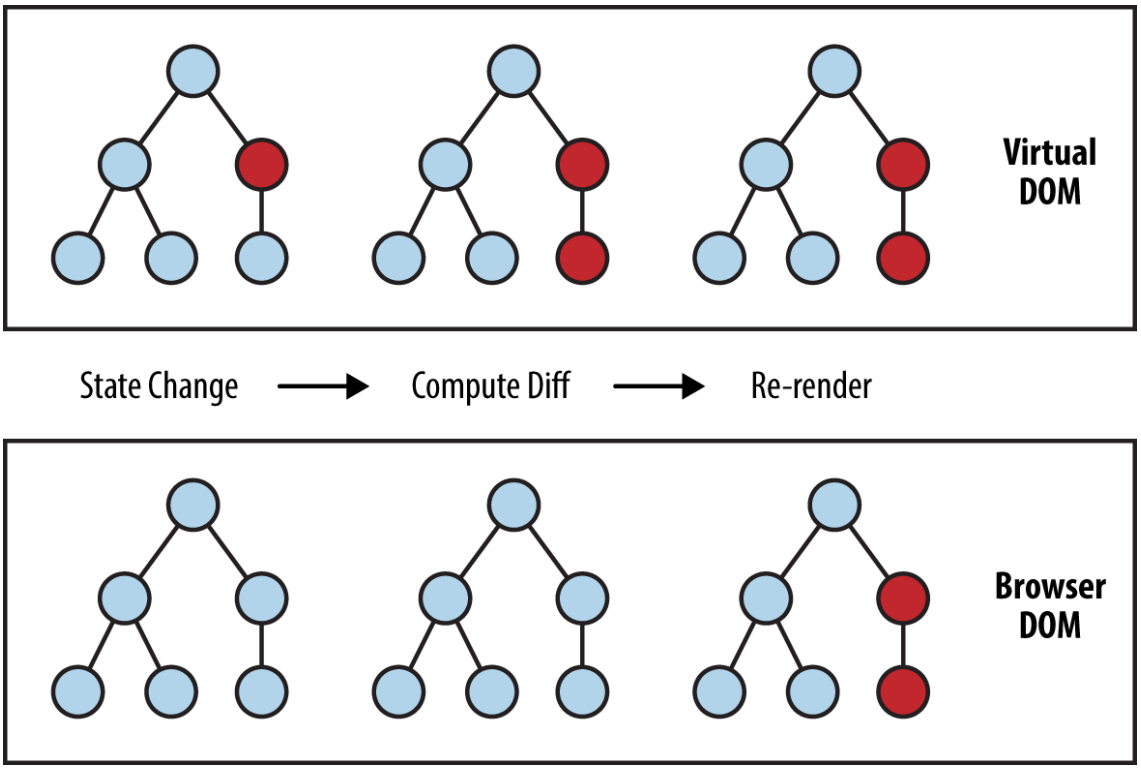
Θα αξιοποιήσουμε κυρίως το built in routing system του καθώς και το default performance optimization του. Σε περίπτωση μελλοντικής επέκτασης της εφαρμογής αυτής το NextJS δίνει τη δυνατότητα κλιμάκωσης (scaling) της εφαρμογής με πολύ πιο γρήγορους ρυθμούς.

4.2.3 Η λογική της τμηματοποίησης στη React.

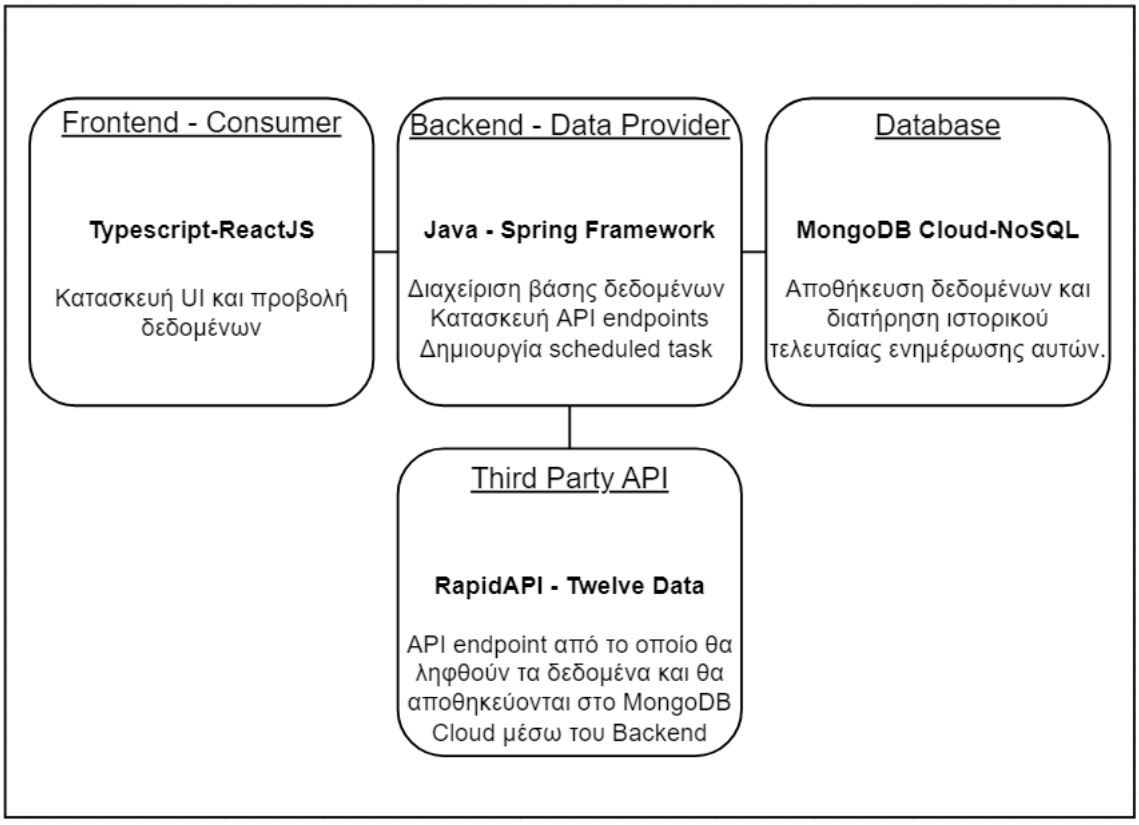
Η React είναι κατάλληλη για εφαρμογές μεγάλης κλίμακας. Αυτό οφείλεται στο γεγονός ότι στον πυρήνα της βασίζεται στην λογική της τμηματοποίησης του κώδικα, της ανεξαρτησίας των τμημάτων κώδικα και αντιμετώπισης τους σαν μεμονωμένα «προγράμματα». Για παράδειγμα στην εφαρμογή της εργασίας αυτής χρειάζεται να υλοποιηθεί search bar σε 2 μέρη (datasets provider & charts). Η λογική υλοποίησης των search bar είναι σχεδόν η ίδια. Συνεπώς με τη React μπορούμε να κατασκευάσουμε ένα search bar component και να το αξιοποιήσουμε σε 2 διαφορετικά μέρη του frontend. Έτσι επιταχύνεται ακόμα περισσότερο το development του frontend της εφαρμογής.

4.2.4 Αποδοτικότητα της React.

Πέρα από τις προοπτικές επεκτασιμότητας, η React έχει πολύ καλύτερες επιδόσεις από την παραδοσιακή μορφή ιστοσελίδων (plain JavaScript). Παραδοσιακά, οι HTML ιστοσελίδες αποτυπώνονται με τη βοήθεια του DOM(document Object Model) (Mozilla MDN Docs, 2022). Ουσιαστικά είναι μια αναπαράσταση της ιστοσελίδας και των τμημάτων αυτής σε ένα δενδροδιάγραμμα. Η καινοτομία της React έγκειται στην τεχνολογία του Virtual DOM, μιας προγραμματιστικής ιδέας όπου οι αλλαγές που γίνονται στην Web εφαρμογή πρώτα αναπαρίστανται πρώτα στο Virtual DOM (Meta Platforms, Inc, n.d.) το οποίο βρίσκεται προσωρινά στη μνήμη. Όταν εντοπιστούν αλλαγές σε σχέση με το πραγματικό DOM (browser DOM) της ιστοσελίδας τότε η React αναλαμβάνει να κάνει update (re-render) την ιστοσελίδα στα συγκεκριμένα σημεία που απαιτείται χωρίς απαραίτητα να διενεργεί ανανέωση ολόκληρης της ιστοσελίδας.



Εικόνα 29: Αναπαράσταση της λειτουργίας του Virtual DOM της React.



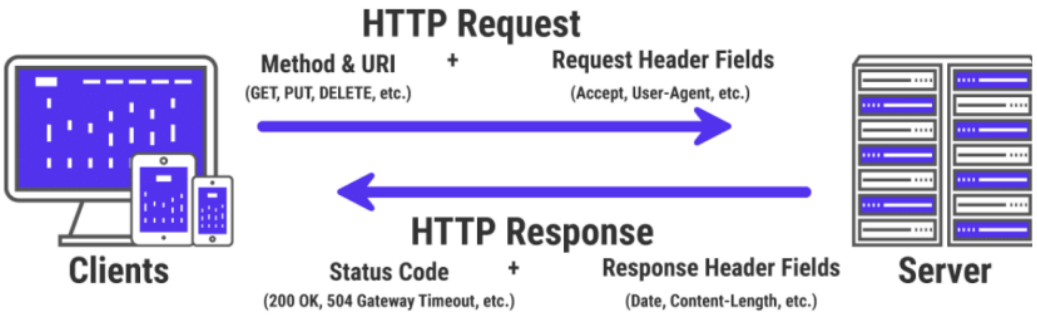
Εικόνα 30: Σύστημα στο οποίο δομείται η εφαρμογή.

4.3 Επικοινωνία backend & frontend.

4.3.1 Hypertext Transfer Protocol (HTTP).

Για την επικοινωνία με χρήση του διαδικτύου αξιοποιείται το πρωτόκολλο Hypertext Transfer Protocol (HTTP) (Wikipedia, n.d.). Πρόκειται για το κύριο Πρωτόκολλο που χρησιμοποιείται από τους browsers για την επικοινωνία με τους servers, με τα πρώτα use cases να εμφανίζονται στην έκδοση 1.1, το έτος 1997. Η βασική ιδέα στηρίζεται στην δομή Request - Response με την προϋπόθεση ύπαρξης φυσικής πρόσβασης μέσα από δίκτυο (καθορίζεται από τους servers και εμφανίζεται με τον όρο origin). Να σημειωθεί ότι οι browsers απαγορεύουν cross origin requests, δηλαδή να λάβουν δεδομένα από servers οι στους οποίους δεν είναι whitelisted, για λόγους ασφαλείας (CORS policy) (Google, n.d.) .Έτσι όταν υπάρχει ανάγκη για πρόσβαση σε δεδομένα ακολουθείται η παρακάτω διαδικασία:

1. Υποβάλλεται HTTP request στον server (συνήθως από browser).
2. Ο server δέχεται το αίτημα και αναλόγως επεξεργάζεται τα δεδομένα που έχει.
3. Ο server επιστρέφει τα δεδομένα μέσα από το δίκτυο.



Εικόνα 31: Διάγραμμα HTTP Request & Response.

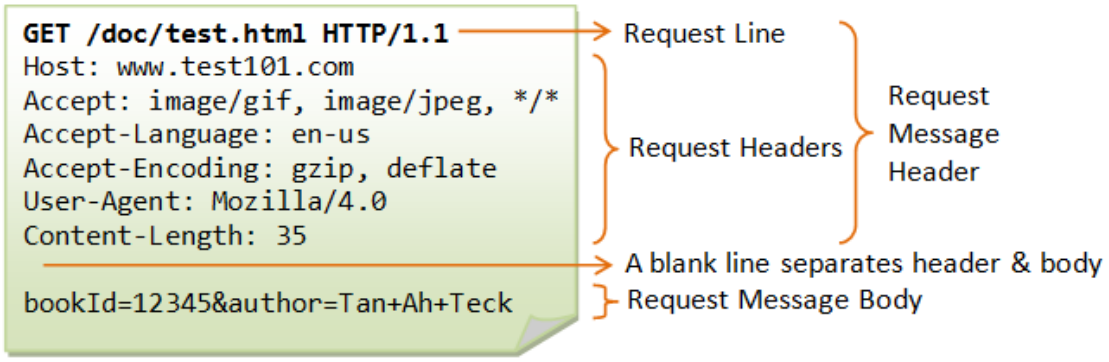
4.3.2 HTTP Request.

Ένα HTTP Request (IBM, 2021) δημιουργείται από έναν client προς ένα server. Σκοπός είναι η πρόσβαση στα δεδομένα – πόρους του server. Το request αυτό αποτελείται από τα παρακάτω χαρακτηριστικά:

1. Request line. Η πρώτη γραμμή του ερωτήματος που καθορίζει τη μέθοδο του request (GET, POST κλπ., θα αναλυθούν παρακάτω), το path του πόρου που ζητείται από τον server καθώς και η έκδοση HTTP. Παράδειγμα: GET /software/htp/cics/index.html HTTP/1.1
2. HTTP headers. Εδώ βρίσκονται πληροφορίες σχετικά με το μήνυμα για τον παραλήπτη σε μορφή name – value pair. Επίσης εμπεριέχονται μεταδεδομένα για το request όπως την γλώσσα του response ή τον τύπο του περιεχομένου (πχ JSON).

   Παράδειγμα: Accept-Language: fr, de

   If-Modified-Since: Fri, 10 Dec 2004 11:22:13 GMT
3. Message body (προαιρετικό). Ενσωματώνει σε μία οντότητα παραμέτρους του request που επιθυμούμε. Παράδειγμα: interval=1day



Εικόνα 32: Παράδειγμα HTTP Request.

Για να υπάρχει ξεκάθαρη εικόνα της πρόθεσης του κάθε request έχουν υλοποιηθεί συγκεκριμένες μέθοδοι που εκφράζουν την ενέργεια που πραγματοποιείται. Συνεπώς υπάρχουν τα παρακάτω HTTP requests:

1. GET, για την ανάκτηση δεδομένων από τον server

2. HEAD, για την αναπαράσταση της πληροφορίας του πόρου χωρίς τα δεδομένα. Αξιοποιείται κυρίως για metadata.
3. POST, για την καταχώρηση του message body στον server.
4. PUT, για τη δημιουργία νέας πληροφορίας στον server.
5. DELETE, για τη διαγραφή δεδομένων από τον server.
6. TRACE, για την επαναποστολή του request για να εντοπιστούν αλλαγές από διαμεσολαβούσες διαδικασίες.
7. OPTIONS, για να πάρουμε πληροφορίες σχετικά με το ποιες μεθόδους υποστηρίζει ο server για το συγκεκριμένο δεδομένο.
8. Patch, για την τροποποίηση – update των πόρων του server.

4.3.3 HTTP Response.

Αφού σταλεί το request στον server συνήθως ακολουθεί και το αντίστοιχο response από αυτόν σχετικά με το πως εξελίχθηκε το αίτημα του client. Για αυτό έχουν εδραιωθεί συγκεκριμένοι κωδικοί από το 100-599 που αναπαριστούν το αποτέλεσμα του request. Συνεπώς, ο server απαντάει στον client με ένα από τα παρακάτω status codes:

1. 1xx Informational
2. 2xx Successful
3. 3xx Redirection
4. 4xx Client Error
5. 5xx Server Error

Τα πιο δημοφιλή status codes είναι 200 για ένα επιτυχές request, 404 για ένα request που δεν βρέθηκε, και 503 που υποδηλώνει τον αποκλεισμό του client από τον server. Το response ακολουθεί ακριβώς την ίδια λογική με το request στο format του, απλά συμπεριλαμβάνει και την απάντηση του server.

```
Request Header:
GET / HTTP/1.1
Host: www.msaleh.co.cc
User-Agent:Mozilla/5.0 (Windows; U; Windows
NT 5.1; en-US; rv:1.9.0.10)
Gecko/2009042316 Firefox/3.0.10
Accept: */*
Connection:close

Response Header:
HTTP/1.1 200 OK
Date: Sat, 09 May 2009 12:27:54 GMT
Server: Apache/2.2.11 (Unix)
Last-Modified: Thu, 12 Feb 2009 15:29:42
GMT
Etag: "c3b-462ba63a46580"-gzip
Cache-Control: max-age=1200, private,
proxy-revalidate, must-revalidate
Expires: Sat, 09 May 2009 12:47:54 GMT
Accept-Ranges: bytes
Content-Length: 976
Content-Type:text/html
```

Εικόνα 33: Παράδειγμα HTTP Response.

5 Συμπεράσματα και μελλοντικές επεκτάσεις.

Συμπερασματικά, με την εργασία αυτή υλοποιήθηκε μία εφαρμογή για την κάλυψη της ανάγκης για την έρευνα των αγορών με σκοπό την επένδυση. Υλοποιήθηκε το Financial Assets Analyst, μια Full Stack Web εφαρμογή, που δίνει τη δυνατότητα δημιουργίας γραφημάτων και δεικτών τεχνικής ανάλυσης χρηματιστηριακών μέσων. Ταυτόχρονα, μέσω της εφαρμογής δίνεται η δυνατότητα λήψης δεδομένων σε μορφή CSV αλλά και μια αυτοματοποιημένη «πρόβλεψη» των τιμών αξιοποιώντας μια στρατηγική τεχνικής ανάλυσης στο backend.

Μέσα από την πτυχιακή εργασία είδαμε πως μπορούμε να αξιοποιήσουμε ένα API για την επικοινωνία του client με το server, τα πρωτόκολλα επικοινωνίας HTTP και πως βοηθούν στην υλοποίηση και ομαλή λειτουργία της εφαρμογής. Τέλος αναλύσαμε τεχνικά θέματα αρχιτεκτονικής τόσο στο frontend όσο και στο backend που ενισχύουν την απόδοση στα πλαίσια μεταφοράς και προβολής δεδομένων τύπου time series.

5.1 Περισσότερα timeframes.

Μια εύκολή, αλλά με κόστος, επέκταση της εφαρμογής είναι να δοθεί η πρόσβαση σε δεδομένα διαφορετικού timeframe από το ημερήσιο. Για να γίνει αυτό, θα πρέπει πρώτα να επεκταθεί το πλάνο στο RapidApi.com που ήδη έχουμε. Δεδομένα χαμηλότερου timeframe απαιτούν πιο συχνές ενημερώσεις, συνεπώς τα API rate limits του free plan θα επηρεάσουν την ομαλή λειτουργία της εφαρμογής.

5.2 User Login / Sign up System.

Μια άλλη μελλοντική επέκταση είναι η δημιουργία ενός login/signup συστήματος. Η ιδέα βασίζεται στην εξατομίκευση της εφαρμογής ανάλογα με τον χρήστη. Αυτό μπορεί να επιτευχθεί αξιόπιστα με τη χρήση ενός συστήματος login/signup.Από εκεί και πέρα οι προοπτικές αυξάνονται με ραγδαίους ρυθμούς.

5.2.1 Watchlist – Notifications.

Εφόσον έχουμε έναν εγγεγραμμένο χρήστη μπορεί να του δίνεται η δυνατότητα να δημιουργεί μια watchlist. Μέσω αυτής θα μπορεί να έχει άμεση πρόσβαση στα χρηματοοικονομικά μέσα που τον ενδιαφέρουν. Μάλιστα, θα μπορεί να κάνει set up ειδοποιήσεις σε περίπτωση που η τιμή περάσει κάποιο σημείο που θα καθορίσει εκείνος. Οι ειδοποιήσεις θα μπορούν να γίνουν μέσω πχ email.

5.2.2 Δυνατότητα trading μέσων άμεσα από την ιστοσελίδα.

Σε συνεργασία με trading platforms θα μπορούσε να προσφέρεται η δυνατότητα αγοράς – πώλησης μέσων μέσα από την εφαρμογή μας. Αυτό μπορεί να επιτευχθεί αξιοποιώντας το API – webhooks της πλατφόρμας trading. Πιθανώς να διαθέτει endpoints για δημιουργία χρηματοοικονομικών εντολών και έτσι να δημιουργηθεί ένα ομαλό user experience έρευνας – τεχνικής ανάλυσης – συναλλαγής.

5.2.3 Δημιουργία Custom Στρατηγικής & automation.

Με την εξατομίκευση των χρηστών μέσω του user login / sign up συστήματος, μπορεί να τους δίνεται η δυνατότητα custom στρατηγικής από το frontend την οποία θα παρακολουθούν. Η ιδέα βασίζεται στο Predictive Analysis που υλοποιήθηκε στην εφαρμογή, αλλά με τη δυνατότητα παραμετροποίησης ανάλογα με τις ανάγκες του κάθε χρήστη.

6 Βιβλιογραφία.

Baeldung. (χ.χ.). *Baeldung*. Ανάκτηση από https://www.baeldung.com/

Gitonga, G. (2021, June 6). *LinkedIn.* Ανάκτηση από Understanding REST architecture: https://www.linkedin.com/pulse/understanding-rest-architecture-gabriel-gitonga/

Google. (2016, March). *mobile site load time statistics*. Ανάκτηση από Think with Google: https://www.thinkwithgoogle.com/consumer-insights/consumer-trends/mobile-site-load-time-statistics/

Google. (χ.χ.). *Google Cloud*. Ανάκτηση από CORS POLICY: https://cloud.google.com/apigee/docs/api-platform/reference/policies/cors-policy#:~:text=Cross%2Dorigin%20resource%20sharing%20(CORS,is%20enforced%20by%20all%20browsers.

Gupta, L. (2022, April 7). *Restful API.* Ανάκτηση από What is REST: https://restfulapi.net/

Harvey, C. R. (2012, October). *candlestick chart*. Ανάκτηση από Financial Glossary: https://financial-dictionary.thefreedictionary.com/candlestick+chart

IBM. (2021, October 13). *IBM*. Ανάκτηση από HTTP requests: https://www.ibm.com/docs/en/cics-ts/5.3?topic=protocol-http-requests

Market capitalization of listed domestic companies (current US$). (1975-2020). Retrieved from https://data.worldbank.org/indicator/CM.MKT.LCAP.CD

Meta Platforms, Inc. (χ.χ.). *ReactJS*. Ανάκτηση από Virtual DOM and Internals: https://reactjs.org/docs/faq-internals.html

Mozilla MDN Docs. (2022, September 29). *Introduction to the Dom*. Ανάκτηση από MDN docs: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

NextJS. (χ.χ.). *NextJs*. Ανάκτηση από https://nextjs.org/

Nielsen Norman Group. (2014, May 4). *Responsive Web Design Definition*. Ανάκτηση από Nielsen Norman Group: https://www.nngroup.com/articles/responsive-web-design-definition/

O'Neil, E. (2008). *Object/relational mapping 2008: hibernate and the entity data model (edm).*

Parker, K., & Fry, R. (2020). More than half of U.S. households have some investment in the stock market. *Pew Research Center*.

Smallcombe, M. (2021, July 23). *Integrate.io.* Ανάκτηση από SQL vs NoSQL: 5 Critical Differences: https://www.integrate.io/blog/the-sql-vs-nosql-difference/

Spring. (χ.χ.). *Spring*. Ανάκτηση από https://spring.io/

Wikipedia. (χ.χ.). *Wikipedia*. Ανάκτηση από Hypertext Transfer Protocol: https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol