Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Σχεδίαση και ανάπτυξη διαδικτυακού καταστήματος με την χρήση VueJS , NuxtJS και Firebase** **Design and development of an online store using VueJS, NuxtJS and Firebase**
Ονοματεπώνυμο Φοιτητή	**Αντώνης Κοτανίδης**
Πατρώνυμο	**Γρηγορίου**
Αριθμός Μητρώου	**ΜΠΣΠ 16013**
Επιβλέπων	**Αλέπης Ευθύμιος, Επίκουρος Καθηγητής**

Ημερομηνία Παράδοσης **Φεβρουάριος 2019**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Αλέπης Ευθύμιος Επίκουρος Καθηγητής	Βίρβου Μαρία Καθηγήτρια	Τσιχριντζής Γεώργιος Καθηγητής

Πίνακας περιεχομένων

Πίνακας Εικόνων

Γλωσσάρι

UML	Unified Modeling Language
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheets
JS	JavaScript

1.Περίληψη

Στη σημερινή εποχή όλο και περισσότερες ιστοσελίδες και εφαρμογές εγκαταλείπουν τις παραδοσιακές βάσεις δεδομένων (σχεσιακές) και χρησιμοποιούν NoSQL βάσεις δεδομένων. Ο κυριότερος λόγος είναι ότι τα τελευταία χρόνια υπάρχει ραγδαία αύξηση του όγκου των δεδομένων τα οποία αλλάζουν γρήγορα και έχουν μεγάλη ποικιλία στη δομή τους, με αποτέλεσμα οι παραδοσιακές βάσεις δεδομένων να μην μπορούν να ανταποκριθούν. Αυτές είναι και οι προκλήσεις που αντιμετωπίζουν οι NoSQL βάσεις δεδομένων, υιοθετώντας διαφορετικές προσεγγίσεις για να αντιμετωπίσουν μεγάλο όγκο δεδομένων, όπως είναι η Firebase.

Έχοντας υπόψιν τις δυνατότητες που μας παρέχουν οι NoSQL βάσεις δεδομένων, αλλά και τις δυνατότητες που που μας προσφέρουν τα JavaScript Frameworks, όπως είναι το VueJS και το NuxtJS, αυτή η μεταπτυχιακή διατριβή επικεντρώνεται στην ανάπτυξη ενός διαδικτυακού καταστήματος με την χρήση Front-end JavaScript Framework και NoSQL βάση δεδομένων. Το διαδικτυακό κατάστημα που θα δημιουργήσουμε, θα χρησιμοποιεί μια cloud βάση δεδομένων την Firebase για την αποθήκευση και την διαχείριση των δεδομένων τις.

2. Abstract

In today's world, more and more web sites and applications are leaving traditional databases (relational) and using NoSQL databases. The main reason is that in recent years there has been a rapid increase in the volume of data that change rapidly and have a wide variety in their structure, resulting in traditional databases not being able to respond. These are the challenges that are facing NoSQL databases, adopting different approaches to address a large amount of data, such as Firebase.

Taking into account the capabilities provided by NoSQL databases and the capabilities offered by JavaScript Frameworks such as VueJS and NuxtJS, this thesis focuses on developing an online store using the Front-end JavaScript Framework and NoSQL database. The online store we create will use a Firebase cloud database to store and manage their data.

3. Εισαγωγή

Τα περισσότερα διαδικτυακά καταστήματα παραδοσιακά χρησιμοποιούν σχεσιακές βάσεις δεδομένων. Τα τελευταία χρόνια με την ραγδαία εξέλιξη των NoSQL βάσεων δεδομένων που μας προσφέρουν μεγάλη ευελιξία, ένα μέρος εξ αυτών τήνει στο να τις χρησιμοποιήσει για την αποθήκευση των δεδομένων τους.

Στη συγκεκριμένη μεταπτυχιακή διατριβή θα επικεντρωθούμε στην ανάπτυξη ενός διαδικτυακού καταστήματος (e-shop) εκτελώντας τις βασικές του λειτουργίες. Θα υπάρχουν διακριτεί ρόλοι ανάμεσα στον διαχειριστή και του απλούς χρήστες. Θα εισάγουμε κατηγορίες προιοντων και προιοντα. Θα χρησιμοποιήσουμε ως βάση δεδομένων την Firebase η οποία είναι cloud βάση δεδομένων και θα δούμε την διάθρωση της ανάλογα με τις λειτουργίες που εκτελέι ο διαχειριστής ή ο χρήστης. Θα εκτελέσουμε ερωτήματα στην βάση (queries) για να κάνουμε αναζήτηση ενός προϊόντος ανάλογα με την κατηγορία στην οποια ανήκει ή με την ονομασία του. Επιπέον θα μπορούμε να κάνουμε ταξινόμηση ενός προ ταξινομήσουμε ανάλογα με την τιμή του. Τέλος θα δούμε πως μπορεί χρήστης να κάνει μια παραγγελία και αυτή να καταχωρηθεί με επιτυχία στην βάση δεδομένων.

Το βασικό κίνητρο για την υλοποίηση του ηλεκτρονικού καταστήματος είναι να εξετάσουμε τις δυνατότητες που μας παρέχει μια NoSQL Cloud βάση δεδομένων για την αποθήκευση των δεδομένων μας και τα JavaScript Frameworks VueJS και NuxtJS για την δημιουργία του. Στο προσεχές μέλλον τα περισσότερα διαδικτυακά καταστήματα θα χρησιμοποιούν NoSQL βάσεις δεδομένων.

4. Θεωρητικό υπόβαθρο

4.1 Hyper Text Markup Language (HTML)

Εικόνα 1. Λογότυπο HTML5

Η HTML (αρχικοποίηση του αγγλικού HyperText Markup Language, ελλ. Γλώσσα Σήμανσης Υπερκειμένου) είναι η κύρια γλώσσα σήμανσης για τις ιστοσελίδες, και τα στοιχεία της είναι τα βασικά δομικά στοιχεία των ιστοσελίδων. Η HTML γράφεται υπό μορφή στοιχείων HTML τα οποία αποτελούνται από ετικέτες (tags), οι οποίες περικλείονται μέσα σε σύμβολα «μεγαλύτερο από» και «μικρότερο από» (για παράδειγμα `<html>`), μέσα στο περιεχόμενο της ιστοσελίδας. Οι ετικέτες HTML συνήθως λειτουργούν ανά ζεύγη (για παράδειγμα `<h1>` και `</h1>`), με την πρώτη να ονομάζεται ετικέτα έναρξης και τη δεύτερη ετικέτα λήξης (ή σε άλλες περιπτώσεις ετικέτα ανοίγματος και ετικέτα κλεισίματος αντίστοιχα). Ανάμεσα στις ετικέτες, οι σχεδιαστές ιστοσελίδων μπορούν να τοποθετήσουν κείμενο, πίνακες, εικόνες κλπ.

Ο σκοπός ενός web browser είναι να διαβάζει τα έγγραφα HTML και τα συνθέτει σε σελίδες που μπορεί κανείς να διαβάσει ή να ακούσει. Ο browser δεν εμφανίζει τις ετικέτες HTML, αλλά τις χρησιμοποιεί για να ερμηνεύσει το περιεχόμενο της σελίδας.

Τα στοιχεία της HTML χρησιμοποιούνται για να κτίσουν όλους του ιστότοπους. Η HTML επιτρέπει την ενσωμάτωση εικόνων και άλλων αντικειμένων μέσα στη σελίδα, ενώ μπορεί να χρησιμοποιηθεί για φόρμες. Παρέχει τις μεθόδους δημιουργίας δομημένων εγγράφων (δηλαδή εγγράφων που αποτελούνται από το περιεχόμενο που μεταφέρουν και από τον κώδικα μορφοποίησης του περιεχομένου) καθορίζοντας δομικά σημαντικά στοιχεία για το κείμενο, όπως κεφαλίδες, παραγράφους, λίστες, συνδέσμους, παραθέσεις και άλλα. Μπορούν επίσης να ενσωματώνονται σενάρια εντολών σε γλώσσες όπως η JavaScript, τα οποία επηρεάζουν τη συμπεριφορά των ιστοσελίδων HTML και από στατικές τις κάνουν διαδραστικές.

Οι Web browsers μπορούν επίσης να αναφέρονται σε στυλ μορφοποίησης CSS για να ορίζουν την εμφάνιση και τη διάταξη του κειμένου και του υπόλοιπου υλικού. Ο οργανισμός W3C, ο οποίος δημιουργεί και συντηρεί τα πρότυπα για την HTML και τα CSS, ενθαρρύνει τη χρήση των CSS αντί διαφόρων στοιχείων της HTML για σκοπούς παρουσίασης του περιεχομένου.

Η HTML5 είναι γλώσσα σήμανσης για τον Παγκόσμιο Ιστό. Είναι η πέμπτη και τελευταία ως σήμερα έκδοση της HTML. Η ομάδα Web Hypertext Application Technology Working Group (WHATWG) άρχισε να εργάζεται για αυτή την έκδοση τον Ιούνιο του 2004 με το όνομα

Web Applications 1.0. Εκδόθηκε τον Οκτώβρη του 2014 από το W3C. Η HTML5 αντικατέστησε τη HTML 4.01, την XHTML 1.0 και την DOM Level 2 HTML. Ο σκοπός είναι η μείωση της ανάγκης για plug-in εμπορικών εταιρειών και πλούσιες διαδικτυακές εφαρμογές (RIA) όπως το Adobe Flash, το Microsoft Silverlight, το Apache Pivot, και η Sun JavaFX.

Η HTML5 αποτελεί φυσική εξέλιξη των προηγούμενων εκδόσεων της HTML και προσπαθεί να καλύψει τις ανάγκες των υπαρχόντων και των μελλοντικών ιστότοπων. Κληρονομεί τη συντριπτική πλειονότητα των λειτουργιών από τους προκατόχους της, γεγονός που σημαίνει ότι αν κάποιος έχει γράψει κώδικα HTML πριν από την έλευση της HTML5, γνωρίζει ήδη αρκετά για την HTML5. Αυτό σημαίνει επίσης ότι μεγάλο μέρος της HTML5 λειτουργεί τόσο σε παλιούς όσο και σε νέους φυλλομετρητές. Η προς τα πίσω συμβατότητα αποτελεί βασική αρχή στον σχεδιασμό της HTML5.

Η ακόλουθη λίστα συνοψίζει τις βασικές αλλαγές στην HTML5:

- Παρουσιάζει αρκετές νέες ετικέτες, που ανταποκρίνονται στο σύγχρονο τρόπο κατασκευής μιας ιστοσελίδας σε ό,τι αφορά τη σελιδοποίηση και τη μορφοποίηση. Οι ετικέτες <article>, <haeder>, <footer>, <nav>, <section>, αντικαθιστούν τα πολλαπλά για την οριοθέτηση τμημάτων διαφορετικού περιεχομένου, τα οποία χρησιμοποιούνταν στις προηγούμενες εκδόσεις.

- Η ετικέτα <canvas> ορίζει μια ορθογώνια περιοχή, που μπορεί να εμφανίσει διάφορα γραφικά σε μια σελίδα, από απλά διαγράμματα μέχρι κινούμενα γραφικά και εξωτερικές εικόνες.

- Δυνατότητα τοπικής αποθήκευσης (offline storage), επιτρέποντας στις ιστοσελίδες να αποθηκεύουν πληροφορίες στον υπολογιστή του χρήστη, έτσι ώστε να μη χρειάζεται να τις λαμβάνουν κάθε φορά που ο χρήστης επισκέπτεται την ιστοσελίδα.

- Δυνατότητα μεταφοράς και απόθεσης, δηλαδή ο χρήστης έχει τη δυνατότητα να μεταφέρει στοιχεία από μια ιστοσελίδα σε άλλη ή από εφαρμογές στον φυλλομετρητή.

- Δυνατότητα αναπαραγωγής ήχου και βίντεο απευθείας, χωρίς την ανάγκη ύπαρξης πρόσθετων προγραμμάτων με τις ετικέτες <audio> και <video>. Οι ετικέτες έχουν κάποιες ιδιότητες/παραμέτρους, οι οποίες ρυθμίζουν τον τρόπο αναπαραγωγής του υλικού.

```
1  <!doctype html>
2  <html>
3      <head>
4          <meta charset="UTF-8">
5          <title>Untitled Document</title>
6      </head>
7
8      <body>
9      </body>
10 </html>
11
```

Εικόνα 2. Παράδειγμα γραφής HTML

4.2 Cascading Style Sheets (CSS)

Εικόνα 3. Λογότυπο CSS

Η CSS (Cascading Style Sheets) είναι μια γλώσσα υπολογιστή που χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός εγγράφου που έχει γραφτεί με μια γλώσσα σήμανσης (πχ. HTML). Η χρήση της CSS επιτρέπει στο έγγραφο να προσαρμόζεται ανάλογα με τη συσκευή που χρησιμοποιείται για την προβολή του όπως για παράδειγμα διαφορετικού μεγέθους οθόνες. Μειώνει τη χρήση bandwidth που απαιτείται για την αποστολή της ιστοσελίδας στον χρήστη και τέλος με την χρήση της δεν υπάρχει η ανάγκη για browser – specific markup και έτσι βελτιώνεται η προσβασιμότητα και αυξάνεται ο πληθυσμός που έχει πρόσβαση στον ιστότοπο.

Στην CSS δεν υπάρχουν εκδόσεις με την συμβατική έννοια· αντί αυτών υπάρχουν επίπεδα, όπου κάθε επίπεδο επεκτείνει, βελτιώνει και προσθέτει στις λειτουργίες του προηγούμενου επιπέδου. Το σύνολο των χαρακτηριστικών κάθε επόμενου επιπέδου είναι ένα υπερσύνολο των λειτουργιών του κατώτερου επιπέδου και με αυτή την λογική ένας περιηγητής συμβατός με ένα ανώτερο επίπεδο θα είναι συμβατός και με κάθε κατώτερο επίπεδο. Το επίπεδο 1 πλέον θεωρείται απαρχαιωμένο και περιττό. Στο επίπεδο 1 ορίζονται τα περισσότερα από τα βασικά στοιχεία όπως ιδιότητες, τιμές και κανόνες.

Στη συνέχεια, εμφανίστηκε το επίπεδο 2 του οποίου η χρήση αποκάλυψε πολλά προβλήματα και έτσι χρειάστηκε να αναθεωρηθεί. Έτσι, προέκυψε η CSS2.1, η οποία αντικατέστησε την CSS 2 και σε οποιαδήποτε περίπτωση υπάρχει ασυμφωνία μεταξύ των δυο προδιαγραφών ισχύει η 2.1. Αν και η 2.1 συνιστά αναθεώρηση της CSS2, κάποια προβληματικά χαρακτηριστικά καταργήθηκαν στην 2.1 και επανεξετάζονται στην CSS3. Η CSS3 επεκτείνει όλα τα επιμέρους τμήματα της CSS2 με την χρήση της CSS2.1 ως βάσης στόχος είναι η ενίσχυση της λειτουργικότητας και η βελτίωση των ορισμών ενώ παραμένει πλήρως συμβατή με την CSS2.

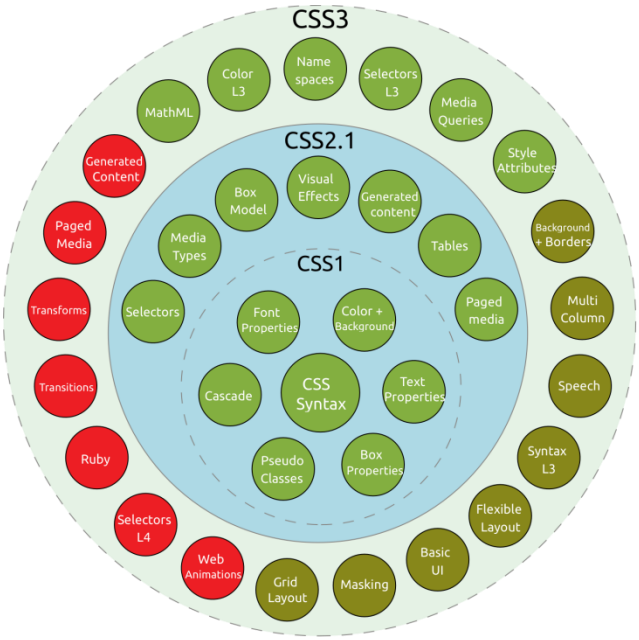
```
Turn this

<style>
  .component-title {}
  .component-title--active {}
</style>

<h1 class="component-title component-title--active">title</h1>

Into this

<style>
  .fdf2 {}
  ._58e4 {}
</style>

<h1 class="fdf2 _58e4">title</h1>
```

Εικόνα 4. Παράδειγμα σύνταξης CSS

4.2.1 CSS 3

Η CSS3 φέρνει πάρα πολλές αλλαγές στο κεφάλαιο "σχεδίαση ιστοσελίδων". Ίσως η σημαντικότερη από αυτές είναι ο διαχωρισμός σε ορίσματα (modules). Ενώ στα προηγούμενα επίπεδα της CSS όλη η εμφάνιση του ιστότοπου καθοριζόταν από ένα ενιαίο κώδικα που καθόριζε τα επιμέρους χαρακτηριστικά, στη CSS3 έχουμε διαχωρισμό του κώδικα σε ορίσματα. Καθένα από αυτά τα ορίσματα έχουν τα δικά τους χαρακτηριστικά και δυνατότητες.

Το όρισμα που έχει προκαλέσει τη μεγαλύτερη επίδραση στους προγραμματιστές είναι τα ερωτήματα μέσων (media queries), το οποίο θα αναλυθεί στο τέταρτο μέρος αυτής της σειράς άρθρων, καθώς είναι αυτό που αποτελεί τον βασικό πυλώνα της responsive σχεδίασης. Συνοπτικά, τα ερωτήματα μέσων επιτρέπουν την εφαρμογή προτάσεων υπό όρους (conditional statements) ώστε υπό διαφορετικές συνθήκες να έχουμε διαφορετικά στυλ που θα προσαρμόζουν τον ιστότοπο στην κάθε οθόνη – ανάλυση.



Εικόνα 5. Ταξινόμηση και κατάσταση των CSS3 Modules

Κάποια από τα νέα χαρακτηριστικά της CSS3 που αξίζει να αναφερθούν, καθώς αποτελούν λύση σε καθημερινά προβλήματα των προγραμματιστών, είναι τα εξής:

- **Rounded Corners:** Στα προηγούμενα επίπεδα της CSS, η χρήση στρογγυλεμένων γωνιών ήταν μια χρονοβόρα διαδικασία καθώς, κάθε φορά, οι προγραμματιστές ήταν υποχρεωμένοι να «κόβουν» τέσσερις κυκλικές εικόνες και να τις τοποθετούν σε κάθε γωνία του div ώστε να παραχθεί το εφέ. Πλέον, οι ιδιότητες «border-radius», «-moz-border-radius» και «-webkit-border-radius» λύνουν τα χέρια των προγραμματιστών, αφού το μόνο που χρειάζεται είναι να οριστεί η ακτίνα του κύκλου.
- **Πολλαπλές εικόνες για φόντο:** Πρακτικά αυτό σημαίνει, πολλά backgrounds σε ένα div. Η CSS3 επιτρέπει τον ορισμό πολλών εικόνων ως φόντο για ένα div, τη θέση του καθενός μέσα στο div και το επίπεδο εμφάνισής τους.
- **Transitions:** Τα CSS3 Transitions είναι ένα σετ κανόνων CSS που επιτρέπουν την αλλαγή τιμών στις ιδιότητες ενός στοιχείου, ομαλά και σταδιακά σε συγκεκριμένο χρόνο. Με άλλα λόγια, επιτρέπουν την αλλαγή τιμών στα στοιχεία, με τέτοιο τρόπο, ώστε το αποτέλεσμα να είναι ένα εφέ σαν animation.

- **Χρώματα:** Η CSS3 παρέχει μια τεράστια ποικιλία των διαθέσιμων χρωμάτων, όπως τα νέα χρωματικά μοντέλα HSL, HSLA και το σημαντικότερο απ' όλα RGBA. Η νέα τιμή «A» στο μοντέλο RGBA αφορά τη διαφάνεια. Μέχρι τώρα για να επιτευχθεί η διαφάνεια, ο προγραμματιστής χρησιμοποιούσε εικόνες με διαφάνεια (.gif, .png). Πλέον, έχει τη δυνατότητα να πειραματιστεί με τη διαφάνεια και να δημιουργήσει ελκυστικά divs εύκολα και γρήγορα.
- **Εφέ κειμένου (text effects) και στις γραμματοσειρές:** Τα εφε λειτουργούν πλέον και στις γραμματοσειρές. Η ιδιότητα «text-shadow» μπορεί να δημιουργήσει οποιουδήποτε πάχους, χρώματος, και σκληρότητας σκιά γύρω από κείμενο. Έως τώρα, ο προγραμματιστής ήταν αναγκασμένος να «βαραίνει» την ιστοσελίδα με εικόνες όταν τα γραφικά ήταν πολύ εντυπωσιακά. Πλέον, με τη χρήση της CSS3, τα παραπάνω εφέ αποδίδονται το ίδιο καλά, όπως θα αποδίδονταν αν γινόταν χρήση εικόνων για την αναπαράσταση τους.
- **font-face:** Με το ερχομό της CSS3 δεν υπάρχει ανάγκη χρήσης των λεγόμενων «web safe fonts» (πχ times new roman), τα οποία μπορεί να είναι ασφαλή, αλλά τις περισσότερες φορές είναι μέτρια σε εμφάνιση. Πλέον μπορούν να χρησιμοποιηθεί μέσα στο αρχείο CSS η ιδιότητα «font-face» και να καλείται η συγκεκριμένη γραμματοσειρά μέσω ενός αρχείου, του αρχείου της γραμματοσειράς το οποίο έχει αποθηκευτεί κάπου στο server.
- **Attribute Matching:** Η ιδιότητα Matching χρησιμοποιείται για να οριστεί στυλ σε tags, τα οποία έχουν κοινούς αρχικούς χαρακτήρες της συμβολοσειράς. Για παράδειγμα η κλάση «newclass» και η κλάση «nextclass» παίρνουν το ίδιο στυλ ορίζοντας απλά ότι το tag τους ξεκινά από «ne».

4.3 JavaScript

Εικόνα 6. Λογότυπο JavaScript

Η JavaScript (JS) είναι διερμηνευμένη γλώσσα προγραμματισμού για ηλεκτρονικούς υπολογιστές. Αρχικά αποτέλεσε μέρος της υλοποίησης των φυλλομετρητών Ιστού, ώστε τα σενάρια από την πλευρά του πελάτη (client-side scripts) να μπορούν να επικοινωνούν με τον χρήστη, να ανταλλάσσουν δεδομένα ασύγχρονα και να αλλάζουν δυναμικά το περιεχόμενο του εγγράφου που εμφανίζεται.

Η JavaScript είναι μια γλώσσα σεναρίων που βασίζεται στα πρωτότυπα (prototypebased), είναι δυναμική, με ασθενείς τύπους και έχει συναρτήσεις ως αντικείμενα πρώτης τάξης. Η σύνταξή της είναι επηρεασμένη από τη C. Είναι γλώσσα βασισμένη σε διαφορετικά προγραμματιστικά παραδείγματα (multi-paradigm), υποστηρίζοντας αντικειμενοστρεφές, προστακτικό και συναρτησιακό στυλ προγραμματισμού.

Η JavaScript χρησιμοποιείται και σε εφαρμογές εκτός ιστοσελίδων - τέτοια παραδείγματα είναι τα έγγραφα PDF, οι εξειδικευμένοι φυλλομετρητές (site-specific browsers) και οι μικρές εφαρμογές της επιφάνειας εργασίας (desktop widgets). Οι νεότερες εικονικές μηχανές και πλαίσια ανάπτυξης για JavaScript (όπως το Node.js) έχουν επίσης κάνει τη JavaScript πιο δημοφιλή για την ανάπτυξη εφαρμογών Ιστού στην πλευρά του διακομιστή (server-side).

```
  addPlace: (city, country) => {
    const id = ++places.length;
    let numType = 'odd';
    if (id % 2) {
      numType = 'even';
    }
    places.push({
      id, city, country, numType,
    });
  },
};

module.exports.addPlace('Mombasa', 'Kenya');
module.exports.addPlace('Kingston', 'Jamaica');
module.exports.addPlace('Cape Town', 'South Africa');
```

Εικόνα 7. Παράδειγμα κώδικα JavaScript

4.3.1 Ιστορία της JavaScript

Η γλώσσα προγραμματισμού JavaScript δημιουργήθηκε αρχικά από τον Brendan Eich της εταιρείας Netscape με την επωνυμία Mocha. Αργότερα, η Mocha μετονομάστηκε σε LiveScript, και τελικά σε JavaScript, κυρίως επειδή η ανάπτυξή της επηρεάστηκε περισσότερο από τη γλώσσα προγραμματισμού Java. LiveScript ήταν το επίσημο όνομα της γλώσσας όταν για πρώτη φορά κυκλοφόρησε στην αγορά σε βήτα (beta) εκδόσεις με το πρόγραμμα περιήγησης στο Web, Netscape Navigator εκδοχή 2.0 τον Σεπτέμβριο του 1995. Η LiveScript μετονομάστηκε σε JavaScript σε μια κοινή ανακοίνωση με την εταιρεία Sun Microsystems στις 4 Δεκεμβρίου, 1995, όταν επεκτάθηκε στην έκδοση του προγράμματος περιήγησης στο Web, Netscape.

Η JavaScript απέκτησε μεγάλη επιτυχία ως γλώσσα στην πλευρά του πελάτη (client-side) για εκτέλεση κώδικα σε ιστοσελίδες, και περιλήφθηκε σε διάφορα προγράμματα περιήγησης στο Web. Η εταιρεία Microsoft ονόμασε την εφαρμογή της JScript για να αποφύγει θέματα εμπορικών σημάτων. Η JScript πρόσθεσε νέους μεθόδους για να διορθώσει τα Y2K-προβλήματα στην JavaScript, οι οποίοι βασίστηκαν στην java.util.Date τάξη της Java. Η JScript περιλήφθηκε στο πρόγραμμα Internet Explorer εκδοχή 3.0, το οποίο κυκλοφόρησε τον Αύγουστο του 1996.

Η JavaScript έχει γίνει μία από τις πιο δημοφιλείς γλώσσες προγραμματισμού ηλεκτρονικών υπολογιστών στον Παγκόσμιο Ιστό (Web). Αρχικά, όμως, πολλοί επαγγελματίες προγραμματιστές υποτίμησαν τη γλώσσα διότι το κοινό της ήταν ερασιτέχνες συγγραφείς ιστοσελίδων και όχι επαγγελματίες προγραμματιστές (μαζί με άλλους λόγους). Με τη χρήση της τεχνολογίας Ajax, η JavaScript γλώσσα επέστρεψε στο προσκήνιο και έφερε πιο επαγγελματική προσοχή προγραμματισμού. Το αποτέλεσμα ήταν ένα καινοτόμο αντίκτυπο στην εξάπλωση των πλαισίων και των βιβλιοθηκών, τη βελτίωση προγραμματισμού με JavaScript, καθώς και αυξημένη χρήση της JavaScript έξω από τα προγράμματα περιήγησης στο Web.

Τον Ιανουάριο του 2009, το έργο CommonJS ιδρύθηκε με στόχο τον καθορισμό ενός κοινού προτύπου βιβλιοθήκης κυρίως για την ανάπτυξη της JavaScript έξω από το πρόγραμμα περιήγησης και μέσα σε άλλες τεχνολογίες (π.χ. server-side).

Εικόνα 8. Η εξέλιξη της JavaScript

4.3.2 Πλεονεκτήματα JavaScript

Η JavaScript προσφέρει τα ακόλουθα πλεονεκτήματα:

- **Επεξεργασία από την Πλευρά του Πελάτη.** Αυτό σημαίνει ότι ο κώδικας εκτελείται από τον επεξεργαστή του χρήστη, αντί του εξυπηρετητή ιστού, εξοικονομώντας έτσι εύρος ζώνης και περιορίζοντας την υπερφόρτωση του εξυπηρετητή.
- **Η εκμάθησή της είναι απλή.** Η σύνταξη αυτής της γλώσσας είναι παρόμοια με τα απλά Αγγλικά, καθιστώντας την εκμάθησή της ευκολότερη για τους προγραμματιστές.
- **Εκτεταμένη Λειτουργικότητα για Ιστοσελίδες.** Οι προσθήκες τρίτων βοηθούν τους προγραμματιστές JavaScript να γράψουν αποσπάσματα κώδικα, τα οποία μπορεί να χρησιμοποιηθούν στις ιστοσελίδες, όπου χρειάζεται.
- **Η Υλοποίησή της είναι απλή.** Η δυνατότητα χρήσης της ίδιας γλώσσας στην κεντρική σελίδα που βλέπει ο χρήστης και το διαχειριστικό τμήμα, καθιστά την εργασία των ομάδων προγραμματισμού ευκολότερη.
- **Οικονομική Γλώσσα.** Δεν απαιτεί κανέναν ειδικό μεταγλωττιστή ή συντάκτη. Το μόνο που χρειάζεστε είναι ένας προγραμματιστής, ένα πρόγραμμα επεξεργασίας κειμένου και έναν περιηγητή για να «τρέξει» τον κώδικα JavaScript.
- **Σχετικά γρήγορη για τον τελικό χρήστη.** Δεν χρειάζεται, πλέον, οι επισκέπτες να συμπληρώσουν μία ολόκληρη φόρμα και να την υποβάλλουν, για να μάθουν πως υπάρχει κάποιο τυπογραφικό λάθος στο πρώτο πεδίο και ότι θα πρέπει να συμπληρώσουν ολόκληρη τη φόρμα ξανά. Με τη JavaScript, κάθε πεδίο μπορεί να επαληθεύεται καθώς συμπληρώνεται από τους χρήστες, γεγονός που παρέχει άμεση ανατροφοδότηση, όταν αυτοί κάνουν κάποιο λάθος.
- **Περιηγητές με ενσωματωμένη JavaScript.** Οι χρήστες του ιστότοπου δεν χρειάζονται ειδικό λογισμικό και λήψεις προγραμμάτων για να δουν τη JavaScript, έτσι κάθε χρήστης έχει την ίδια εμπειρία.

4.3.3 Μοντέλο εκτέλεσης

Η αρχική έκδοση της Javascript βασίστηκε στη σύνταξη στη γλώσσα προγραμματισμού C, αν και έχει εξελιχθεί, ενσωματώνοντας πια χαρακτηριστικά από νεότερες γλώσσες. Αρχικά χρησιμοποιήθηκε για προγραμματισμό από την πλευρά του πελάτη (client), που ήταν ο φυλλομετρητής (browser) του χρήστη, και χαρακτηρίστηκε σαν client-side γλώσσα προγραμματισμού. Αυτό σημαίνει ότι η επεξεργασία του κώδικα Javascript και η παραγωγή του τελικού περιεχομένου HTML δεν πραγματοποιείται στο διακομιστή, αλλά στο πρόγραμμα περιήγησης των επισκεπτών, ενώ μπορεί να ενσωματωθεί σε στατικές σελίδες HTML. Αντίθετα, άλλες γλώσσες όπως η PHP εκτελούνται στο διακομιστή (server-side γλώσσες προγραμματισμού).Η χρήση της Javascript στο διακομιστή εμφανίζεται πάλι σήμερα, με τη διάδοση του Node.js, ενός μοντέλου προγραμματισμού βασισμένο στα γεγονότα (events).

5.Web Services

5.1 Ορισμός

Τα web services είναι μια τεχνολογία που επιτρέπει στις εφαρμογές να επικοινωνούν μεταξύ τους ανεξαρτήτως πλατφόρμας και γλώσσας προγραμματισμού. Ένα web service είναι μια διεπαφή λογισμικού (software interface) που περιγράφει μια συλλογή από λειτουργίες οι οποίες μπορούν να προσεγγιστούν από το δίκτυο μέσω πρότυπων μηνυμάτων XML. Χρησιμοποιεί πρότυπα βασισμένα στη γλώσσα XML για να περιγράψει μία λειτουργία (operation) προς εκτέλεση και τα δεδομένα προς ανταλλαγή με κάποια άλλη εφαρμογή. Μια ομάδα από web services οι οποίες αλληλεπιδρούν μεταξύ τους καθορίζει μια εφαρμογή web services.

Η Microsoft μέσα από το MSDN της καταλήγει ότι όλα τα web services έχουν τρία (3) κοινά χαρακτηριστικά:

- Τα web services εκθέτουν χρήσιμη λειτουργικότητα σε χρήστες του διαδικτύου μέσα από ένα πρότυπο δικτυακό πρωτόκολο. Στις περισσότερες περιπτώσεις αυτό το πρωτόκολλο είναι το SOAP (Simple Object Access Protocol).
- Τα web services παρέχουν ένα τρόπο να περιγράψουν τις διεπαφές τους με αρκετή λεπτομέρεια ώστε να επιτρέψουν στο χρήστη τους να χτίσει μια εφαρμογή πελάτη η οποία να επικοινωνήσει μαζί τους. Η περιγραφή συνήθως παρέχεται σε ένα έγγραφο XML το οποίο ονομάζεται έγγραφο WSDL (Web Services Description Language).
- Τα web services παρέχουν ένα τρόπο να περιγράψουν τις διεπαφές τους με αρκετή λεπτομέρεια ώστε να επιτρέψουν στο χρήστη τους να χτίσει μια εφαρμογή πελάτη η οποία να επικοινωνήσει μαζί τους. Η περιγραφή συνήθως παρέχεται σε ένα έγγραφο XML το οποίο ονομάζεται έγγραφο WSDL (Web Services Description Language).

5.2 Πλεονεκτήματα σε σχέση με παλαιότερες κατανεμημένες τεχνολογίες

5.2.1 Ευκολότερος χειρισμός δεδομένων

Παραδοσιακά το κυριότερο πρόβλημα στις κατανεμημένες τεχνολογίες ήταν το λεγόμενο tight-coupling ή στα ελληνικά η ισχυρή συνδεδιμότητα. Μια εφαρμογή που καλούσε μια άλλη απομακρυσμένη ήταν αυστηρά δεμένη με αυτή από την κλήση λειτουργίας (function call) που εκτελούσε και τις παραμέτρους που περνούσε. Στα περισσότερα συστήματα πριν από την έλευση των web services ο τρόπος επικοινωνίας ήταν μια σταθερή διεπαφή με λίγη έως καθόλου ευελεξία ή προσαρμοστικότητα στα περιβάλλοντα ή τις ανάγκες που μεταβάλλονται συνεχώς.

Τα web services χρησιμοποιούν τη γλώσσα XML η οποία μπορεί να περιγράψει οποιαδήποτε δεδομένα σε ένα πραγματικά ανεξάρτητο από πλατφόρμα τρόπο για ανταλλαγή αυτών των δεδομένων μεταξύ συστημάτων. Με αυτόν τον τρόπο οδηγούμαστε σε εφαρμογές με χαλαρή συνδεσιμότητα (loosely-coupled). Επιπλέον τα web services μπορούν να λειτουργήσουν σε πιο αφηρημένο επίπεδο στο οποίο μπορούν να επαναξιολογήσουν, να τροποποιήσουν ή να χειριστούν τύπους δεδομένων δυναμικά κατά περίπτωση. Έτσι σε τεχνικό επίπεδο τα web services μπορούν να χειριστούν δεδομένα πολύ ευκολότερα και να επιτρέψουν στο λογισμικό να επικοινωνεί πιο ελεύθερα.

5.2.2 Απλότητα πρωτοκόλλου επικοινωνίας

Τα web services χρησιμοποιούν ως πρωτόκολλο επικοινωνίας το SOAP. Το πρωτόκολλο αυτό είναι πολύ πιο απλό από πρωτόκολλα παλαιότερων τεχνολογιών όπως αυτά που χρησιμοποιούνταν από τα κατανεμημένα περιβάλλοντα CORBA , DCOM, RPC. Έτσι το να δημιουργήσει κανείς μια υλοποίηση SOAP που υπόκειται στα πρότυπα (standards-compliant) είναι πολύ πιο εύκολο. Σήμερα μπορεί να βρει κανείς υλοποιήσεις του SOAP από τις μεγαλύτερες εταιρίες πληροφορικής αλλά ακόμη και από μεμονωμένους προγραμματιστές, πράγμα αδιανόητο για παλαιότερες κατανεμημένες τεχνολογίες

5.2.3 Απλότητα υποδομής

Τα web services λειτουργούν με πρότυπες γλώσσες και πρωτόκολλα όπως η XML , το HTTP και το TCP/IP. Η πλειονότητα των εταιριών έχουν ήδη την δικτυακή υποδομή και τους ανθρώπους με γνώσεις και εμπειρία που τη συντηρούν. Έτσι το κόστος για την εφαρμογή των web services είναι σημαντικά μικρότερο από αυτό των προηγούμενων τεχνολογιών.

5.2.4 Ευκολία στην επικοινωνία

Με τις προηγούμενες τεχνολογίες η συνεργασία μεταξύ εταιριών ήταν ένα θέμα διότι κατανεμημένες τεχνολογίες όπως CORBA και DCOM χρησιμοποιούσαν μη πρότυπες πόρτες. Σαν αποτέλεσμα η συνεργασία σήμαινε άνοιγμα "οπών" στα τείχη προστασίας (firewalls) κάτι που πολλές φορές δεν ήταν αποδεκτό από τους ανθρώπους της πληροφορικής σε μια εταιρία αφού έθετε σε κίνδυνο στην ασφάλεια των συστημάτων. Το γεγονός αυτό δεν επέτρεπε δυναμική συνεργασία λόγω του ότι απαιτούσε μια χειροκίνητη διαδικασία για τη συνεργασία μιας εταιρίας με τους συνεργάτες της. Τα web services μπορούν να χρησιμοποιήσουν (μεταξύ άλλων) το HTTP ως πρωτόκολλο μεταφοράς και τα περισσότερα τείχη προστασίας επιτρέπουν την πρόσβαση μέσω της θύρας 80 (πρότυπη θύρα για το HTTP). Με αυτόν τον τρόπο οδηγούμαστε σε ευκολότερες και δυναμικές συνεργασίες μεταξύ των συστημάτων των εταιριών.

5.2.5 Διαλειτουργικότητα και ευκολία ανάπτυξης νέων εφαρμογών

Οι προηγούμενες κατανεμημένες τεχνολογίες υπέφεραν από ζητήματα διαλειτουργικότητας διότι κάθε προμηθευτής (vendor) υλοποιούσε το δικό του πρότυπο για distributed object messaging. Με την XML σαν το μόνο πρότυπο στα web services, συστήματα φτιαγμένα από διαφορετικές τεχνολογίες όπως η Java και το .Net μπορούν να επικοινωνήσουν μεταξύ τους 4. Επιπλέον λόγω της απλότητας της XML είναι πολύ πιο εύκολο να γραφτούν νέες εφαρμογές σε μικρό χρονικό διάστημα.

5.3 Τα web services από την επιχειρηματική σκοπιά

Σε ένα υψηλότερο εννοιολογικό επίπεδο, μπορούμε να δούμε τα web services σαν μονάδες εργασίας (units of work). Ένα βήμα παραπέρα, αυτές οι μονάδες μπορούν να συνδυαστούν σε εργασίες επιχειρησιακού προσανατολισμού για να χειριστούν συγκεκριμένες επιχειρησιακές λειτουργίες. Αυτό με τη σειρά του επιτρέπει σε μη τεχνικούς ανθρώπους να σχεδιάσουν εφαρμογές οι οποίες μπορούν να χειριστούν τα επιχειρησιακά ζητήματα συνδυάζοντας τα web services σε ροές εργασίας. Σε μία αναλογία από τον χώρο των αυτοκινήτων, ο αρχιτέκτονας των επιχειρησιακών διαδικασιών μπορεί να συνδυάσει ολόκληρο τον κινητήρα με το πλαίσιο, τη μετάδοση και τα υπόλοιπα υποσυστήματα παρά να ασχοληθεί με τα εσωτερικά κομμάτια του

κινητήρα. Επιπλέον η δυναμική πλατφόρμα σημαίνει ότι ο κινητήρας μπορεί να συνεργαστεί με τη μετάδοση ή άλλα υποσυστήματα διαφορετικών κατασκευαστών.

Αυτό που προκύπτει από αυτή την πτυχή είναι ότι τα web services βοηθούν στη γεφύρωση του κενού που υπάρχει ανάμεσα στους ανθρώπους του επιχειρείν και τους ανθρώπους της πληροφορικής σε ένα οργανισμό. Οι άνθρωποι του επιχειρείν μπορούν να περιγράψουν γεγονότα και δραστηριότητες και οι τεχνικοί μπορούν να τα συνδέσουν με τις κατάλληλες υπηρεσίες.

Με καθολικά καθορισμένες διεπαφές και καλά σχεδιασμένες λειτουργίες, γίνεται επίσης εύκολο να επαναχρησιμοποιηθούν αυτές οι λειτουργίες άρα και οι εφαρμογές που αντιπροσωπεύουν. Η επαναχρησιμοποίηση μιας εφαρμογής λογισμικού σημαίνει καλύτερη απόδοσης της επένδυσης (return on investment) διότι μπορεί να παράγει περισσότερα με τους ίδιους πόρους. Επιτρέπει στους ανθρώπους να εξετάσουν το ενδεχόμενο χρησιμοποίησης μιας ήδη υπάρχουσας εφαρμογής με ένα διαφορετικό τρόπο ή το ενδεχόμενο προσφοράς αυτής σε ένα συνεργάτη με διαφορετικό τρόπο, αυξάνοντας ενδεχομένως τις επιχειρησιακές συναλλαγές μεταξύ των συνεργατών. Επομένως, τα κύρια ζητήματα που τα web services προσπαθούν να λύσουν είναι τα ζητήματα ολοκλήρωσης (integration) δεδομένων και εφαρμογών και αυτά της μεταμόρφωσης των τεχνικών λειτουργιών σε υπολογιστικές λειτουργίες επιχειρησιακού προσανατολισμού.

5.4 Τα web services από την τεχνική σκοπιά

Ενώ τα web services προσφέρουν όλα αυτά τα δυναμικά χαρακτηριστικά ώστε να συνδυάσουμε υπηρεσίες σε εφαρμογές, πρέπει πρώτα να χτίσουμε αυτές τις υπηρεσίες. Οι γλώσσες προγραματισμού στην πληροφορική συνεχώς εξελίσονται. Ξεκινήσαμε δεκαετίες πριν με την ιδέα της function στην οποία παρέχουμε μερικές παραμέτρους, εκτελεί μια λειτουργία με αυτές τις παραμέτρους, και επιστρέφει μια τιμή βασισμένη στους υπολογισμούς που έγιναν. Τελικά, αυτή η πρώτη έννοια εξελίχθηκε σε αντικείμενο (object) όπου κάθε αντικείμενο είχε όχι απλώς έναν αριθμό από λειτουργίες (functions) που μπορεί να εκτελέσει αλλά και τις δικές του ιδιωτικές μεταβλητές (private data variables), αντί να στηρίζεται σε εξωτερικές μεταβλητές του συστήματος που προηγουμένως έκανα πιο περίπλοκη την ανάπτυξη εφαρμογών. Δεδομένου ότι οι εφαρμογές άρχισαν να επικοινωνούν μεταξύ τους, η έννοια του καθορισμού καθολικών διεπαφών (universal interfaces) για αντικείμενα έγινε σημαντική, επιτρέποντας αντικείμενα σε διαφορετικές πλατφόρμες να επικοινωνούν ακόμη και αν είχαν αναπτυχθεί σε διαφορετικές γλώσσες προγραμματισμού και εκτελούνταν σε διαφορετικά λειτουργικά συστήματα.

Στο πιο πρόσφατο βήμα, τα web services προχώρησαν μπροστά με την έννοια των διεπαφών και επικοινωνιών καθορισμένων με XML, ενώνοντας τελικά κάθε είδους εφαρμογή με οποιαδήποτε άλλη, όπως και παρέχοντας την ελευθερία στις εφαρμογές αν αλλάξουν και να εξελιχθούν με το χρόνο, αρκεί να είναι σχεδιασμένες σύμφωνα με την κατάλληλη διεπαφή. Η μεταβλητότητα της XML είναι αυτό που κάνει τα web services διαφορετικά από τεχνολογίες προηγούμενων γενεών. Επιτρέπει το διαχωρισμό της γραμματικής δομής (syntax) και της γραμματικής έννοιας (semantics), και του πώς αυτά υποβάλλονται σε επεξεργασία και κατανοούνται από μία υπηρεσία και το περιβάλλον μέσα στο οποίο υπάρχει. Έτσι λοιπόν τώρα, τα αντικείμενα μπορούν να καθοριστούν σαν υπηρεσίες, οι οποίες επικοινωνούν με άλλες υπηρεσίες σε γραμματική καθορισμένη σε XML, με την οποία κάθε υπηρεσία μεταφράζει και αναλύει το μήνυμα σύμφωνα με μην τοπική της υλοποίησης και το περιβάλλον της. Κατά συνέπεια μια δικτυακή εφαρμογή μπορεί πραγματικά να συντεθεί από πολλαπλές οντότητες διαφόρων

υλοποιήσεων και σχεδιασμών εφόσον προσαρμόζονται στους κανόνες που καθορίζονται από την προσανατολισμένη στις υπηρεσίες αρχιτεκτονική τους.

Κατά συνέπεια, με αυτά στο μυαλό, τα web services μας επιτρέπουν:

- Την αλληλεπίδραση μεταξύ υπηρεσιών σε οποιαδήποτε πλατφόρμα, γραμμένες σε οποιαδήποτε γλωσσα προγραμματισμού.
- Να αντιληφθούμε λειτουργίες εφαρμογών ως εργασίες, οδηγούμενοι σε ανάπτυξη και ροές εργασιών προσανατολισμένες σε εργασίες. Αυτό επιτρέπει μια υψηλότερη αφαίρεση του λογισμικού το οποίο μπορεί να υιοθετηθεί από λιγότερο τεχνικά καταρτισμένους χρήστες.
- Τη χαλαρή συνδεσιμότητα μεταξύ εφαρμογών, πράγμα που σημαίνει ότι αλληλεπιδράσεις μεταξύ υπηρεσιών δε θα χαλάνε κάθε φορά που υπάρχει κάποια αλλαγή το πώς μία ή περισσότερες υπηρεσίες σχεδιάζονται ή υλοποιούνται.
- Την προσαρμογή ήδη υπάρχουσων εφαρμογών στις μεταβαλλόμενες επιχειρησιακές συνθήκες και ανάγκες των πελατών.
- Να παρέχουμε υπάρχουσες εφαρμογές λογισμικού με διαπαφές υπηρεσιών χωρίς να αλλάξουμε τις αρχικές εφαρμογές, επιτρέποντάς τους να λειτουργούν πλήρως στο περιβάλλον των υπηρεσιών.
- Να παρέχουμε υπάρχουσες εφαρμογές λογισμικού με διαπαφές υπηρεσιών χωρίς να αλλάξουμε τις αρχικές εφαρμογές, επιτρέποντάς τους να λειτουργούν πλήρως στο περιβάλλον των υπηρεσιών.

5.5 Εφαρμογές των web services

Τα πρώτα web services σκόπευαν να είναι πηγές πληροφορίας τις οποίες μπορεί κανείς πολύ εύκολα να ενσωματώσει στις εφαρμογές του : τιμές μετοχών, προβλέψεις καιρού, αποτελέσματα αθλητικών παινιδιών κλπ. Έιναι εύκολο να φανταστεί κανείς μια ολόκληρη κατηγορία εφαρμογών που μπορεί να κατασκευάσει ώστε να αναλύει και να συνδυάζει πληροφορία που τον ενδιαφέρει και να την παρουσιάζει με ποικίλους τρόπους. Για παράδειγμα, θα μπορούσαμε να έχουμε ένα λογιστικό φύλλο (spreadsheet) το οποίο συνοψίζει όλη την οικονομική μας εικόνα : μετοχές, τραπεζικούς λογαριασμούς, δάνεια κλπ. Αν αυτή η πληροφορία ήταν διαθέσιμη μέσω web services το λογιστικό φύλλο θα μπορούσε να ενημερώνεται συνεχώς. Οι περισσότερες από αυτές τις πληροφορίες είναι ήδη διαθέσιμες στον παγκόσμιο ιστό αλλά τα web services θα κάνουν την προγραμματιστική πρόσβαση σε αυτές πιο εύκολη και πιο αξιόπιστη.

Εκθέτοντας ήδη υπάρχουσες εφαρμογές σαν web services θα επιτρέψει στους χρήστες να κατασκευάσουν νέες πιο ισχυρές εφαρμογές οι οποίες χρησιμοποιούν τα web services σαν δομικά στοιχεία. Για παράδειγμα, ένας χρήστης θα μπορούσε να αναπτύξει μια εφαρμογή προμηθειών η οποία να παίρνει αυτόματα τιμές από προμηθευτές, να επιστρέπει στο χρήστη να επιλέξει προμηθευτή, να υποβάλει την παραγγελία και να παρακολουθεί την αποστολή έως ότου να γίνει η παραλαβή της. Η εφαρμογή του προμηθευτή, εκτός από το να εκθέτει τις υπηρεσίες της στον ιστό, θα μπορούσε να χρησιμοποιήσει άλλα web services για να ελέγξει την πιστωληπτική ικανότητα του πελάτη, να χρεώσει τον τραπεζικό λογαριασμό του πελάτη και να καθορίσει την αποστολή με μια εταιρία μεταφορών.

6.JavaScript Framework

6.1 VueJS

Εικόνα 9.Λογότυπο VueJS

Το Vue.js δημιουργήθηκε για να οργανώσει και να απλοποιήσει την ανάπτυξη εφαρμογών ιστού. Το έργο επικεντρώνεται στο να γίνουν πιο προσιτά οι ιδέες για την ανάπτυξη του UI στο διαδίκτυο (components, declarative UI, hot-reloading, time-travel debugging, κλπ). Προσπαθεί να είναι λιγότερο δογματικό και έτσι πιο εύκολο για τους προγραμματιστές να το υιοθετήσουν. Διαθέτει μια προοδευτικά υιοθετήσιμη αρχιτεκτονική. Η κεντρική βιβλιοθήκη επικεντρώνεται στη δηλωτική απόδοση και τη σύνθεση συστατικών και μπορεί να ενσωματωθεί σε υπάρχουσες σελίδες. Οι προηγμένες λειτουργίες που απαιτούνται για σύνθετες εφαρμογές όπως η δρομολόγηση, η διαχείριση κατάστασης της σελίδας και η κατασκευή εργαλείων προσφέρονται μέσω επίσημα διατηρούμενων βιβλιοθηκών και πακέτων υποστήριξης.

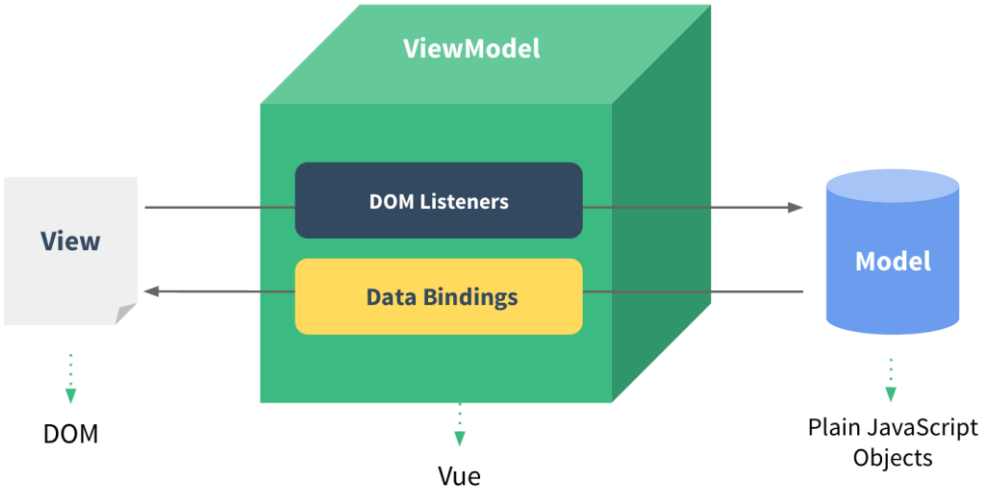
Τα πλεονεκτήματά του είναι πολλά. Έχει ως στόχο να χρησιμοποιείται ως ένα progressive framework στο οποίο κομμάτια μπορούν να προστεθούν ανά πάσα στιγμή. Αρχικά, μπορεί να φορτώσει μέσω CDN και χρησιμοποιείται για τη δημιουργία απλών components. Κατά την ανάπτυξή του προστίθενται νέες λειτουργίες, όπως router, state management, http abstraction κ.λ.π, αλλά τίποτα από αυτά δεν πρέπει να γίνει εξ' αρχής.

Οπότε έχει πολύ μικρό learning curve, αφού δεν χρειάζεται να μάθει κάποιος ένα ολόκληρο framework προκειμένου να λύσει μικρά προβλήματα. Μπορεί ο καθένας να ξεκινήσει με μικρή λειτουργικότητα για την λύση απλών προβλημάτων και να προσθέσει πολυπλοκότητα όταν αυτή χρειαστεί. Επιπλέον, το VueJS είναι γρήγορο. Αν εξαιρέσουμε το React με το Virtual DOM, το VueJS έχει κάνει εξαιρετική δουλειά στο optimization.

6.1.1 Δέσμευση ενεργών δεδομένων (Reactive Data Binding)

Στον πυρήνα του Vue.js είναι ένα σύστημα αντιδραστικής δέσμευσης δεδομένων που καθιστά εξαιρετικά απλή τη διατήρηση των δεδομένων μας και του DOM σε συγχρονισμό. Όταν χρησιμοποιούμε το jQuery για χειροκίνητο χειρισμό του DOM, ο κώδικας που γράφουμε είναι συχνά επιτακτικός, επαναλαμβανόμενος και επιρρεπής σε σφάλματα. Το Vue.js περιλαμβάνει την έννοια της προβολής δεδομένων. Με απλά λόγια, σημαίνει ότι χρησιμοποιούμε ειδική σύνταξη στα κανονικά πρότυπα HTML για να "δεσμεύσουμε" το DOM στα υποκείμενα δεδομένα. Μόλις δημιουργηθούν οι δεσμεύσεις, τότε το DOM θα διατηρηθεί σε συγχρονισμό με τα δεδομένα. Κάθε φορά που τροποποιείτε τα δεδομένα,

ενημερώνεται το DOM ανάλογα. Ως αποτέλεσμα, το μεγαλύτερο μέρος της λογικής εφαρμογής μας τώρα χειρίζεται άμεσα τα δεδομένα, αντί να ανακατεύει με ενημερώσεις DOM.



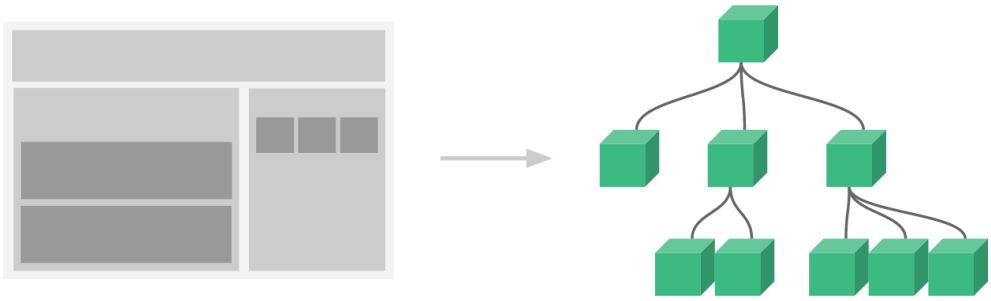
Εικόνα 10.Σύστημα VueJS

```
<!-- this is our View -->
<div id="example-1">
  Hello {{ name }}!
</div>
// this is our Model
var exampleData = {
  name: 'Vue.js'
}
// create a Vue instance, or, a "ViewModel"
// which links the View and the Model
var exampleVM = new Vue({
  el: '#example-1',
  data: exampleData
})
```

Εικόνα 11.Παράδειγμα κώδικα σε VueJS

6.1.2 Σύστημα στοιχείων (Component System)

Το Σύστημα Στοιχείων (Component System) είναι μια άλλη σημαντική ιδέα στο Vue.js, μας επιτρέπει να χτίζουμε εφαρμογές μεγάλης κλίμακας που αποτελούνται από μικρά, αυτοδύναμα και συχνά επαναχρησιμοποιούμενα components.



Εικόνα 12.Σύστημα στοιχείων σε VueJS

Στην πραγματικότητα, μια τυπική μεγάλη εφαρμογή που κατασκευάζεται με το Vue.js θα σχηματίζει ακριβώς αυτό που δείχνει η εικόνα - ένα δέντρο από components.

6.2 Nuxt.js

Εικόνα 13.Λογότυπο NuxtJS

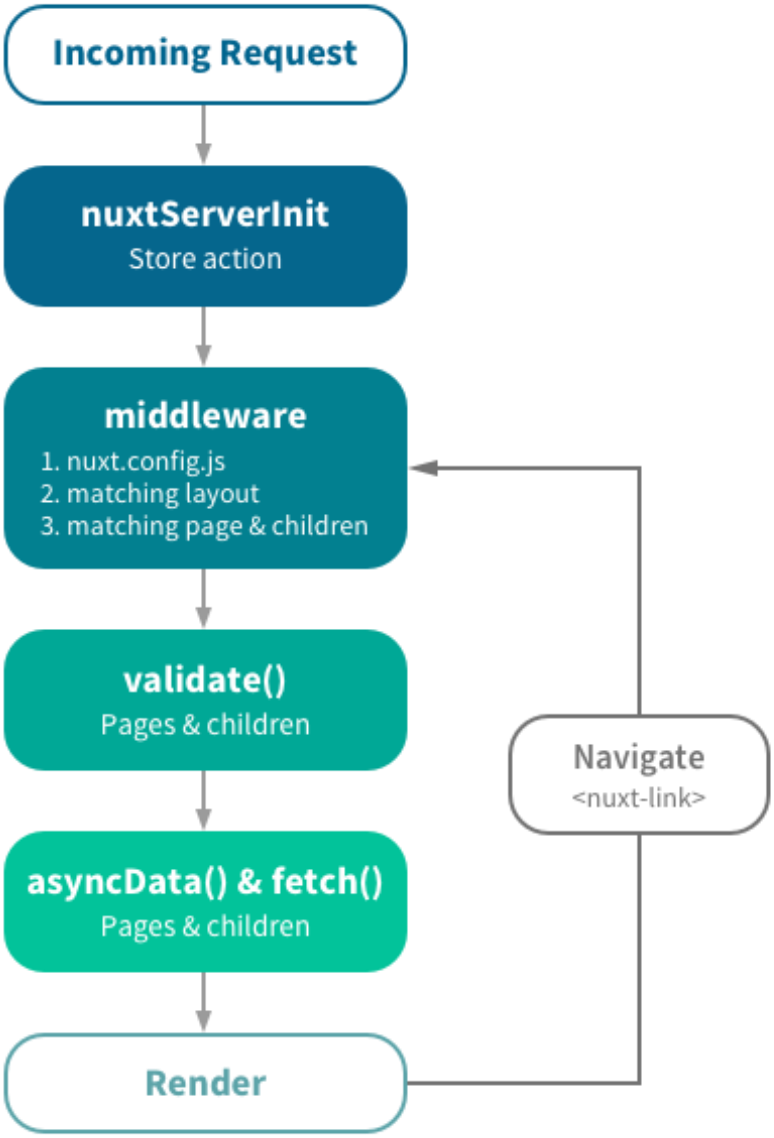
Το κύριο πεδίο εφαρμογής του Nuxt.js είναι η απόδοση του User Interface (UI), ενώ παράλληλα αφαιρεί τη διανομή πελάτη / διακομιστή (client/server). Το Nuxt.js προρρυθμίζει όλες τις ρυθμίσεις που απαιτούνται για να καταστήσει την ανάπτυξη ενός διακομιστή εφαρμογών Vue.js πιο ευχάριστη. Επίσης παρέχει μια επιλογή ανάπτυξης που ονομάζεται: nuxt generate, το οποίο είναι το επόμενο μεγάλο βήμα για την ανάπτυξη Web εφαρμογών με microservices. Επιπλέον μπορούμε να χρησιμοποιήσουμε το Nuxt.js για να δημιουργήσουμε γρήγορες εφαρμογές μιας σελίδας (single page application). Ως framework το Nuxt.js έρχεται με πολλές δυνατότητες για να μας βοηθήσει στην ανάπτυξη μεταξύ της πλευράς του πελάτη και της πλευράς του διακομιστή, όπως Asynchronous Data, Middleware και Layouts.

6.2.1 Χαρακτηριστικά Nuxt.js

Τα χαρακτηριστικά που εμπεριέχει το Nuxt.js είναι τα ακόλουθα:

- Μπορούμε να γράψουμε αρχεία .vue
- Αυτόματος διαχωρισμός κώδικα
- Ισχυρό σύστημα δρομολόγησης με ασύγχρονα δεδομένα
- Στατική αρχειοθέτηση αρχείων
- Συμπλήρωση και οριοθέτηση του JS & CSS
- Διαχείριση στοιχείου <head> (<title>, <meta>, κ.λπ.)
- Προ-επεξεργαστής: Sass, Less, Stylus κ.λπ.
- Επέκταση με αρθρωτή αρχιτεκτονική

Αυτό το σχήμα δείχνει τι καλείται από το Nuxt.js όταν καλείται ο διακομιστής ή όταν ο χρήστης πλοηγείται μέσω της εφαρμογής <nuxt-link>:



Εικόνα 14.Σχήμα NuxtJS

6.3 NodeJS

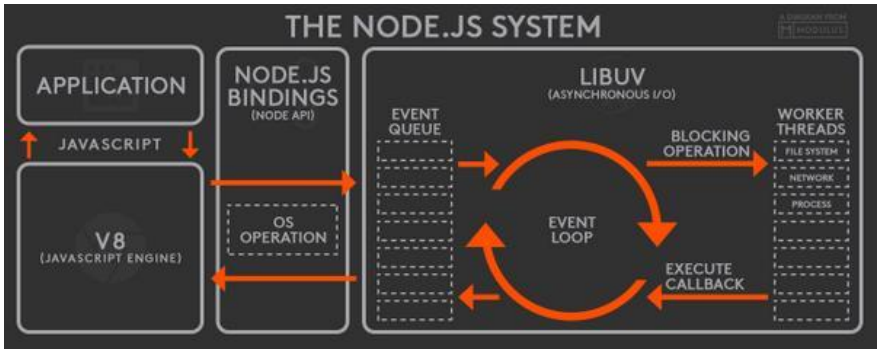
Εικόνα 15.Λογότυπο NodeJs

Το Node.js είναι μια πλατφόρμα ανάπτυξης λογισμικού (κυρίως διακομιστών) χτισμένη σε περιβάλλον Javascript. Στόχος του Node είναι να παρέχει ένα εύκολο τρόπο δημιουργίας κλιμακωτών διαδικτυακών εφαρμογών. Σε αντίθεση από τα περισσότερα σύγχρονα περιβάλλοντα ανάπτυξης εφαρμογών δικτύων μία διεργασία node δεν στηρίζεται στην πολυνηματικότητα αλλά σε ένα μοντέλο ασύγχρονης επικοινωνίας εισόδου/εξόδου.

Το Node.js υιοθετεί μια διαφορετική προσέγγιση για την υλοποίηση των εισερχόμενων και εξερχόμενων νημάτων από τον διακομιστή. Εκτελεί έναν βρόχο συμβάντων με μια συγκεκριμένη διαδικασία που έχει καταχωρηθεί με το σύστημα για την διαχείριση συνδέσεων και κάθε νέα σύνδεση προκαλεί την πυροδότηση μιας λειτουργίας επανάκλησης JavaScript. Αυτή μπορεί να χειριστεί I/O αιτήματα με μη αποκλειστικές κλήσεις I/O και αν είναι απαραίτητο μπορεί να κρίνει ώστε να εκτελέσει λειτουργίες αποκλεισμού ή εντατικής χρήσης CPU και να ισορροπήσει φορτία μεταξύ πυρήνων της CPU όπου είναι αναγκαίο. Η προσέγγιση αυτή απαιτεί λιγότερη μνήμη για να χειριστεί περισσότερες συνδέσεις σε σχέση με τις περισσότερες ανταγωνιστικές αρχιτεκτονικές που κλιμακώνονται με νήματα, συμπεριλαμβανομένου του Apache HTTP Server, των διάφορων Java Server, των ISS και ASP.NET και του γνωστού Ruby on Rails.

Το Node χαρακτηρίζεται από την έμφαση στην ασύγχρονη επικοινωνία μεταξύ των υπολογιστικών πόρων. Αυτό επιτυγχάνεται με την χρήση συμβάντων (events) που προσφέρει η Javascript και ονομάζονται callbacks. Για παράδειγμα όταν ένας περιηγητής ιστού φορτώσει πλήρως ένα αρχείο, ένας χρήστης πατάει κάποιο κουμπί, ολοκληρώνεται ένα αίτημα AJAX, τα συμβάντα αυτά πυροδοτούν ένα συγκεκριμένο callback. Αυτό με την σειρά του επιτρέπει την ροή του κώδικα χωρίς να αφήνει ανενεργό τον επεξεργαστή προκειμένου να εκτελεστεί μια λειτουργία, όπως μια επιτυχής ανάγνωση αρχείου από τον δίσκο.

Η κοινότητα έχει δημιουργήσει ένα ολόκληρο οικοσύστημα από βιβλιοθήκες που προορίζονται ή είναι συμβατές με το node. Ανάμεσά τους εργαλεία που ξεχώρισαν όπως το Mongodb, το MVC framework Express και τη βιβλιοθήκη για επικοινωνία σε πραγματικό χρόνο Socket.IO παίζουν σημαντικό ρόλο υποστηρίζοντας την ασύγχρονη διάδραση με τις παραδοσιακές και NoSQL μεθόδους βάσεων δεδομένων. Αυτό επιτυγχάνεται με την χρήση του node package manager το οποίο επιτρέπει την εγκατάσταση των παραπάνω βιβλιοθηκών. Χρησιμοποιείται συνήθως σε εφαρμογές Chat, Proxy, Http Server καθώς και για παρακολούθηση εφαρμογών και του συστήματος (monitoring).



Εικόνα 16.Σύστημα NodeJS

6.3.1 Ιστορία

Το Node.js δημιουργήθηκε από τον Ryan Dahl το 2009. Η δημιουργία και η συντήρηση του έργου χορηγήθηκε από την εταιρία Joyent. Η ιδέα για την ανάπτυξη του node προήλθε από την ανάγκη του Ryan Dahl να βρει τον πιο αποδοτικό τρόπο να ενημερώνει τον χρήστη σε πραγματικό χρόνο για την κατάσταση ενός αρχείου που ανέβαζε στο διαδίκτυο. Επίσης επηρεάστηκε από το Mongrel του Zed Shaw. Επιπροσθέτως μετά από αποτυχημένα έργα σε C, Lua, Haskell η κυκλοφορία της μηχανής V8 (V8 JavaScript Engine) της Google τον ώθησε να ασχοληθεί με την Javascript.

Το 2011, ένας διαχειριστής πακέτων εισήχθη στη βιβλιοθήκη Node.js, με το όνομα npm. Ο διαχειριστής πακέτων επιτρέπει τη δημοσίευση και τον διαμοιρασμό των ανοιχτού κώδικα βιβλιοθηκών Node.js από την κοινότητα, και είναι σχεδιασμένος να απλοποιεί την εγκατάσταση, αναβάθμιση και απεγκατάσταση των βιβλιοθηκών. Το πρώτο Node.js που κατασκευάστηκε για να υποστηρίζει τα Windows κυκλοφόρησε τον Ιούλιο του 2011.

6.3.2 Χαρακτηριστικά

Το runtime περιβάλλον του **Node.js** χρησιμοποιεί τη μηχανή V8 JavaScript της Google (Google V8 JavaScript Engine) για την ερμηνεία του JavaScript. Με τη συγκεκριμένη πλατφόρμα γίνεται η υλοποίηση μιας ποικιλίας εργαλείων server και εφαρμογών.

Βασικά χαρακτηριστηκά του Node.js:

- **Ασύγχρονο και Event Driven.** Όλα τα API της βιβλιοθήκης του Node.js είναι ασύγχρονα, δηλαδή, non-blocking. Αυτό σημαίνει ότι ένας server βασισμένος σε Node.js δεν περιμένει ένα API να για να επιστρέψει δεδομένα. Ο server μεταφέρετε στο επόμενο API μετά την κλήση του και ο μηχανισμός κοινοποίησης των Event του Node.js βοηθά το server να πάρει την απάντηση από την προηγούμενη κλήση που έκανε στο API.

- **Πολύ γρήγορο.** Λόγω του ότι είναι βασισμένο στη μηχανή V8 JavaScript της Google η βιβλιοθήκη του εκτελεί πολύ γρήγορα τον κώδικα.

- **No Buffering.** Οι Node.js εφαρμογές ποτέ δεν κάνουν buffer τα δεδομένα.

6.3.3 Επισκόπηση

Το Node.js επιτρέπει τη δημιουργία διαδικτυακών διακομιστών και εργαλείων δικτύωσης που χρησιμοποιούν την JavaScript και μία συλλογή από modules που χειρίζονται διάφορες βασικές λειτουργίες. Τα modules χειρίζονται το σύστημα αρχείων εισόδου/εξόδου, τη δικτύωση (HTTP, TCP, UDP, DNS, ή TLS/SSL), δυαδικά δεδομένα (ενδιάμεσες μνήμες), λειτουργίες

κρυπτογράφησης, ροές δεδομένων και άλλες βασικές λειτουργίες. Τα modules του Node χρησιμοποιούν ένα API που είναι σχεδιασμένο να μειώνει την πολυπλοκότητα της συγγραφής εφαρμογών διακομιστών. Τα πλαίσια μπορούν να χρησιμοποιηθούν για να επιταχύνουν την ανάπτυξη εφαρμογών και κοινών πλαισίων όπως τα Express.js, Socket.IO και Connect.

Το Node.js αρχικά χρησιμοποιήθηκε για την κατασκευή προγραμμάτων δικτύου, όπως διαδικτυακών διακομιστών, κάτι που το καθιστούσε παρόμοιο με την PHP και την Python. Η μεγαλύτερη διαφορά μεταξύ της PHP και του Node.js είναι ότι η PHP είναι μια blocking γλώσσα, όπου οι εντολές εκτελούνται αφού πρώτα έχει ολοκληρωθεί η προηγούμενη εντολή, ενώ το Node.js είναι μια non-blocking γλώσσα όπου οι εντολές εκτελούνται παράλληλα, και χρησιμοποιεί ανακλήσεις για να σηματοδοτήσει κάποια ολοκλήρωση.

7. Βάσεις δεδομένων

7.1 NoSQL Βάσεις Δεδομένων

Τα τελευταία χρόνια έχει παρατηρηθεί μια ραγδαία αύξηση στη δημοτικότητα μιας οικογένειας τεχνολογιών αποθήκευσης δεδομένων, γνωστών και ως NoSQL (ακρωνύμιο του Not Only SQL). Αλλά ο όρος NoSQL χαρακτηρίζει περισσότερο τι δεν είναι – δεν είναι SQL-κεντρικές σχεσιακές βάσεις δεδομένων – παρά τι στην ουσία είναι αυτές οι βάσεις δεδομένων.

Ιστορικά οι περισσότερες διαδικτυακές εφαρμογές τρέχουν πάνω σε σχεσιακές βάσεις δεδομένων. Την τελευταία δεκαετία όμως τα δεδομένα που αντιμετωπίζουμε είναι πολύ μεγαλύτερα σε μέγεθος, αλλάζουν γρηγορότερα, και έχουν μεγάλη ποικιλία στη δομή τους, κάνοντας τα παραδοσιακά RDBMS συστήματα δύσκολο να ανταποκριθούν. Αυτές είναι και οι προκλήσεις που αντιμετωπίζουν οι NoSQL βάσεις δεδομένων.

Ο χρόνος εκτέλεσης των ερωτημάτων (queries) στις σχεσιακές βάσεις δεδομένων αυξάνεται, καθώς αυξάνεται το μέγεθος των πινάκων και ο αριθμός των ενώσεων (joins) που εκτελούνται. Αυτό δεν είναι λάθος των ίδιων των βάσεων, αλλά του τρόπου που αντιμετωπίζουν τα δεδομένα καθώς για να απαντήσουν σε ένα ερώτημα αρχικά βρίσκουν όλα τα πιθανά αποτελέσματα και στην συνέχεια τα φιλτράρουν. Αντίθετα οι NoSQL βάσεις δεδομένων υιοθετούν διαφορετικές προσεγγίσεις για να αντιμετωπίσουν μεγάλο όγκο δεδομένων, προκειμένου να αποφύγουν τις πολλές ενώσεις.

Εκτός από το μεγάλο τους μέγεθος, τα σημερινά δεδομένα συχνά αλλάζουν πολύ γρήγορα. Αυτός ο υψηλός ρυθμός αλλαγής των δεδομένων έχει υψηλό υπολογιστικό κόστος στις σχεσιακές βάσεις δεδομένων, καθώς για να γίνει μια αλλαγή πρέπει να προσπελάσουμε πολλούς πίνακες.

Πέρα όμως από τον υψηλό ρυθμό αλλαγής των ίδιων των δεδομένων, μπορεί να αλλάξει ακόμα και η δομή τους. Με άλλα λόγια, εκτός από την τιμή μιας συγκεκριμένης ιδιότητας μπορεί να αλλάξει και η ίδια η δομή των στοιχείων που έχουν αυτή την ιδιότητα. Οι σχεσιακές βάσεις δεδομένων έχουν αυστηρά καθορισμένο σχήμα, και απαιτούν τον αυστηρό ορισμό αυτού του σχήματος πριν οποιοδήποτε δεδομένο αποθηκευτεί σε αυτές. Η αλλαγή του σχήματος, αφού μπουν δεδομένα στην βάση, απαιτεί μεγάλο λειτουργικό κόστος και είναι μια διαδικασία που συχνά αποφεύγεται. Αντίθετα οι NoSQL βάσεις είναι απελευθερωμένες από αυτόν τον περιορισμό, καθώς δεν έχουν κάποιο προκαθορισμένο σχήμα (είναι schema-free). Έτσι επιτρέπουν στους προγραμματιστές εύκολα να ενσωματώνουν νέους τύπους δεδομένων ώστε να εμπλουτίζουν τις εφαρμογές τους.

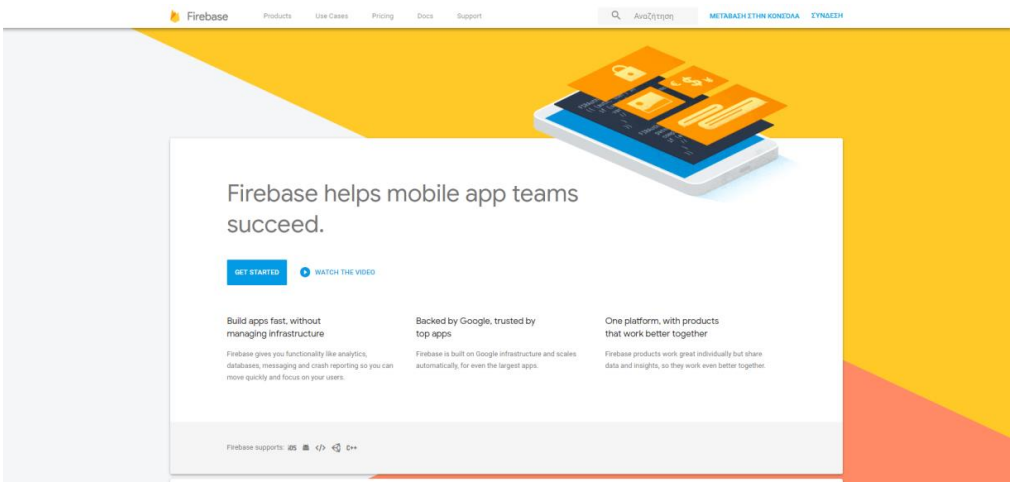
Οι NoSQL βάσεις δεδομένων χωρίζονται σε 4 κατηγορίες με βάση τον τρόπο που μοντελοποιούν τα δεδομένα. Είναι οι Document Stores, οι Key-Value Stores, οι Column Family και οι βάσεις δεδομένων γράφου (graph databases).

7.2 Firebase

Εικόνα 17.Λογότυπο Firebase

Η Firebase είναι μια βάση δεδομένων πραγματικού χρόνου, η οποία, με τα APIs που παρέχει για διάφορες γλώσσες και πλατφόρμες. Επιτρέπει στους σχεδιαστές λογισμικού να αποθηκεύουν και να συγχρονίζουν τα δεδομένα μιας εφαρμογής μεταξύ πολλών χρηστών. Κυκλοφόρησε τον Σεπτέμβριο του 2011, από τους Andrew Lee και Lames Tamplin, ενώ το 2014 αγοράστηκε από την Google. Με τα χρόνια, η Firebase εξελίχθηκε σε μια πλήρη σουίτα εργαλείων για ανάπτυξη εφαρμογών, προσθέτοντας πολλές λειτουργίες στον πυρήνα της, ορισμένες από τις οποίες είναι οι εξής:

- Firebase Analytics: εφαρμογή που γνωστοποιεί την χρήση μίας εφαρμογής αλλά και τον χρόνο ενασχόλησης του χρήστη με την εφαρμογή
- Firebase Auth: υπηρεσία που προσφέρει αυθεντικοποίηση χρηστών. Υποστηρίζει και τη δυνατότητα αυθεντικοποίησης μέσω κοινωνικών δικτύων (Facebook, GitHub, Twitter και Google), ενώ παρέχει και σύστημα διαχείρισης χρηστών όπου οι σχεδιαστές μπορούν να έχουν πλήρη έλεγχο των χρηστών.
- Firebase Storage: παρέχει ασφαλές «ανέβασμα» (upload) και «κατέβασμα» (download) αρχείων στην εφαρμογή και μπορεί να αποθηκεύει όλο το υλικό (εικόνες, ήχους, βίντεο) που παράγει ο οποιοσδήποτε χρήστης.



Εικόνα 18.Αρχική οθόνη Firebase

Για να αποκτήσει κάποιος πρόσβαση στην βάση δεδομένων και στις υπόλοιπες υπηρεσίες της Firebase, πρέπει να έχει δημιουργήσει ένα λογαριασμό στην mail υπηρεσία της Google (Gmail). Η βάση της Firebase την οποία χρησιμοποιεί ο οποιοσδήποτε διαχειριστής βρίσκεται σε cloud servers της Google, γι' αυτό άλλωστε και η πρόσβαση γίνεται μέσω internet. Τα δεδομένα είναι

αποθηκευμένα σε μορφή JSON (JavaScript Object Notation), η οποία τα διαρθρώνει δενδρικά και δεν ακολουθούν τους κανόνες των SQL βάσεων δεδομένων. Ένα μεγάλο πλεονέκτημα της είναι ότι περιλαμβάνει και offline λειτουργία, κυρίως για εφαρμογές Android. Όταν δεν υπάρχει δυνατότητα σύνδεσης με το Internet, η Firebase αποθηκεύει τα δεδομένα σε μία τοπική μνήμη (cache memory) και μόλις η σύνδεση επανέλθει, αμέσως στέλνει τα νέα δεδομένα στην βάση, ενημερώνοντας με συγχρονισμό και τους άλλους χρήστες της εφαρμογής.

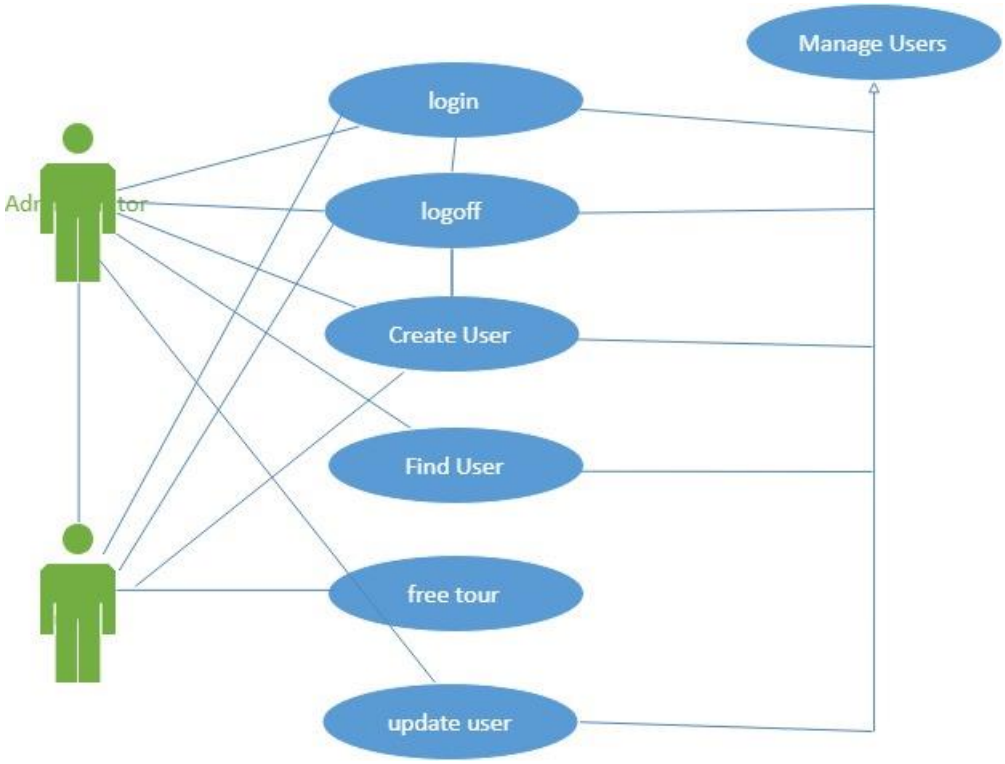
8. Προδιαγραφές

8.1 Σχεδιασμός UML Διαγράμματα

Για τον σχεδιασμό της εφαρμογής χρησιμοποιήσαμε UML διαγράμματα τα οποία είναι μία κοινώς αποδεκτή πρακτική για την περιγραφή του υπό ανάπτυξη λογισμικού. Κατά την φάση του σχεδιασμού χρησιμοποιήσαμε τα αντίστοιχα UML διαγράμματα για να περιγράψουμε την δομή και την λειτουργικότητα της εφαρμογής που σκοπεύουμε να υλοποιήσουμε.

8.1.1 Διάγραμμα Περιπτώσεων Χρήσης

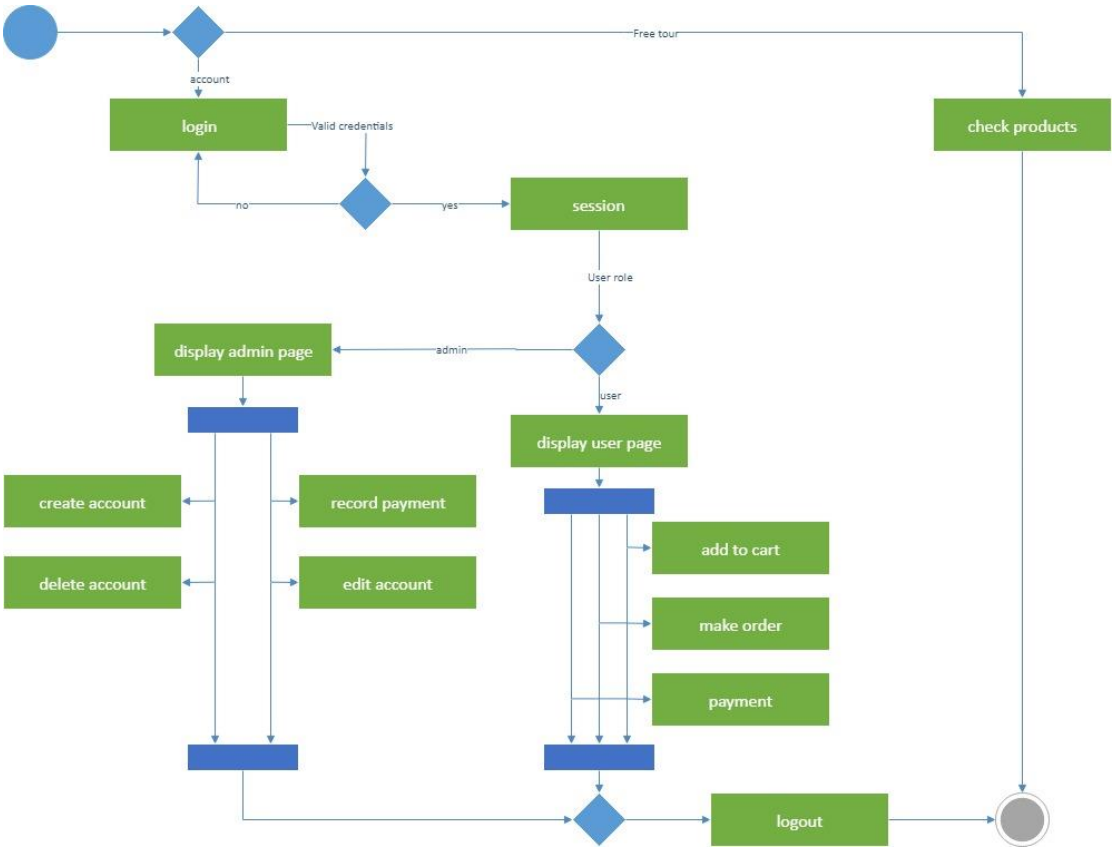
Όπως φαίνεται και στο διάγραμμα, οι χρήστες διακρίνονται σε διαχειριστές και απλούς χρήστες. Όλοι έχουν τη δυνατότητα εισόδου και εξόδου απ το σύστημα. Ο διαχειριστής μπορεί να δημιουργεί ή να διαγράφει έναν λογαριασμό, ενώ ο χρήστης μόνο να δημιουργεί. Ο απλώς χρήστης μπορεί να πλοηγηθεί χωρίς είσοδο στο σύστημα, ενώ ο administrator μπορεί να δει τα στοιχεία όλων των χρηστών που έχουν λογαριασμό.



Εικόνα 19.Διάγραμμα Περιπτώσεων Χρήσης

8.1.2 Διάγραμμα δραστηριοτήτων

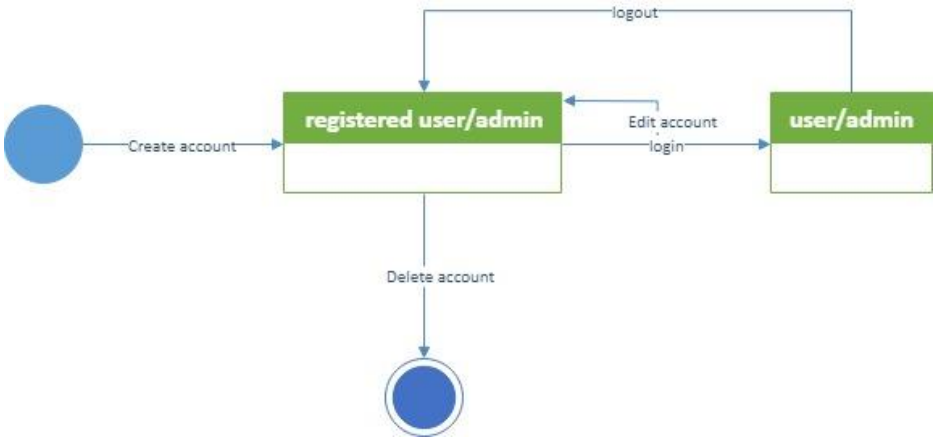
Το διάγραμμα δραστηριοτήτων περιγράφει τη ροή εργασιών μέσα στο σύστημα.



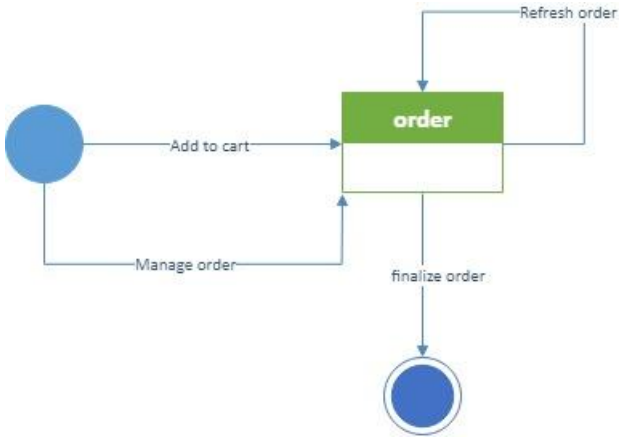
Εικόνα 20.Διάγραμμα Δραστηριοτήτων

8.1.3 Διαγράμματα καταστάσεων

Τα διαγράμματα καταστάσεων χρησιμοποιούνται για τη μοντελοποίηση λειτουργιών από συμβάντα, περιγράφοντας όλες τις καταστάσεις που μπορεί να βρίσκεται ένα αντικείμενο και τον τρόπο που αλλάζει η κατάστασή τους.



Εικόνα 21.Διαγράμματα καταστάσεων διαχείρισης λογαριασμού



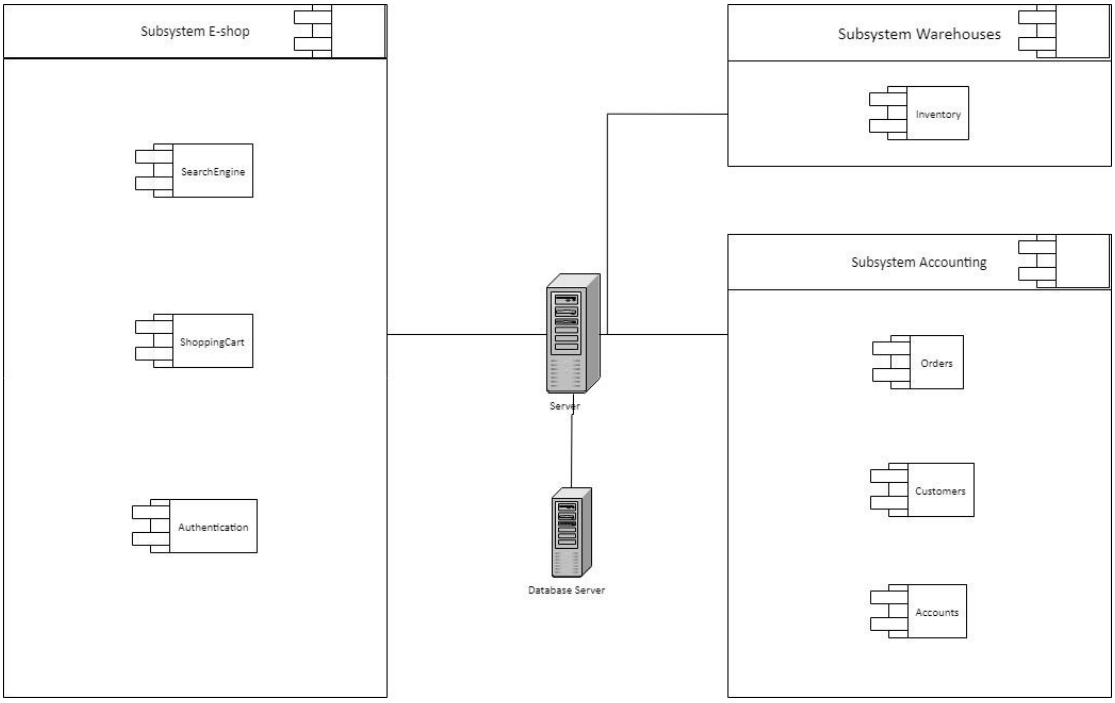
Εικόνα 22.Διαγράμματα καταστάσεων διαχείρισης παραγγελίας

8.1.4 Διάγραμμα Εξαρτημάτων

Το διάγραμμα εξαρτημάτων είναι ένα διάγραμμα δομής υλοποίησης που χρησιμοποιείται για να μοντελοποιήσει τον Πηγαίο κώδικα, τις Εκτελέσιμες εκδόσεις, τις Βάσεις δεδομένων και Δυναμικά προσαρμοζόμενα συστήματα

Περιλαμβάνει:

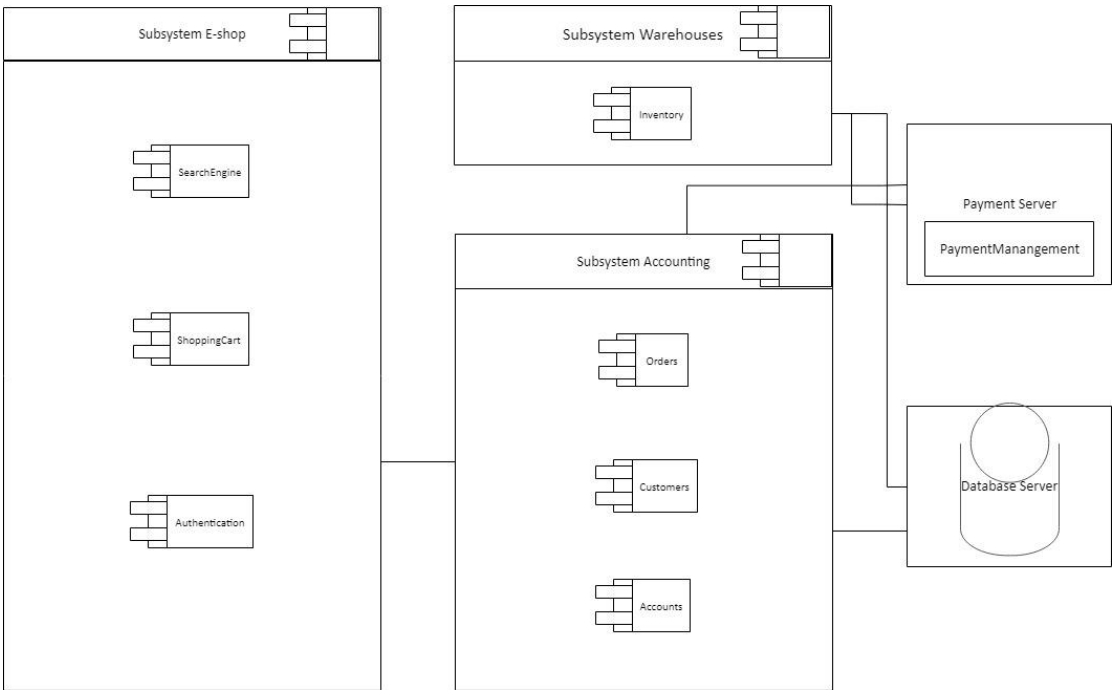
- Εξαρτήματα
- Διεπαφές (*interfaces*)
- Σχέσεις εξάρτησης, γενίκευσης, σύνδεσης και υλοποίησης



Εικόνα 23.Διάγραμμα Εξαρτημάτων

8.1.5 Διάγραμμα διανομής

Τα διαγράμματα διανομής δείχνουν μία όψη των κόμβων και όλων των συστατικών που τρέχουν σε αυτούς. Τα διαγράμματα διανομής αναπαριστούν το υλικό που είναι βασισμένο το σύστημα, το λογισμικό και τον τρόπο διασύνδεσης των κόμβων.



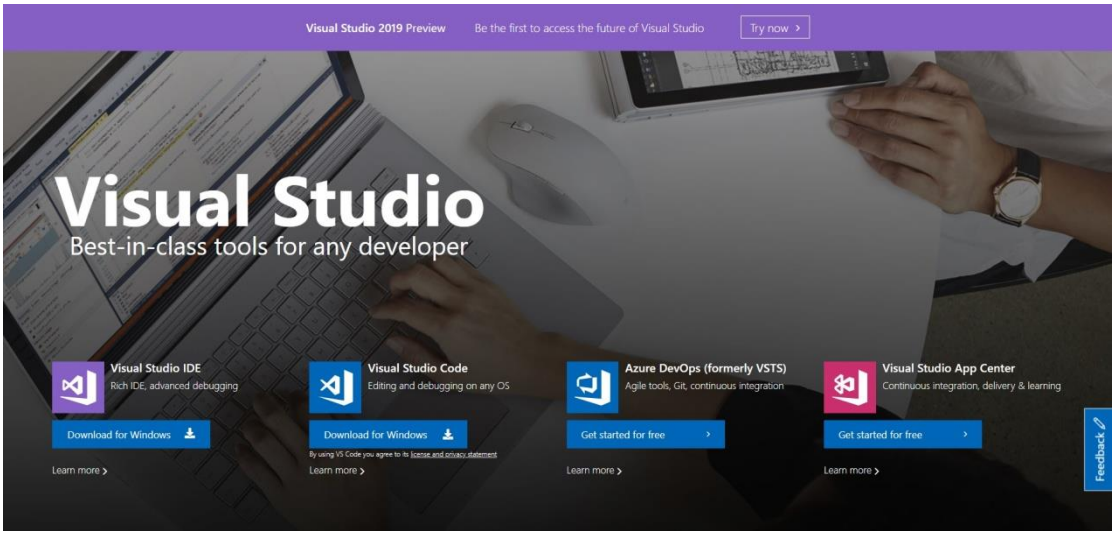
Εικόνα 24.Διάγραμμα διανομής

9. Υλοποίηση Ηλεκτρονικού Καταστήματος

9.1 Προετοιμασία Υλοποίησης

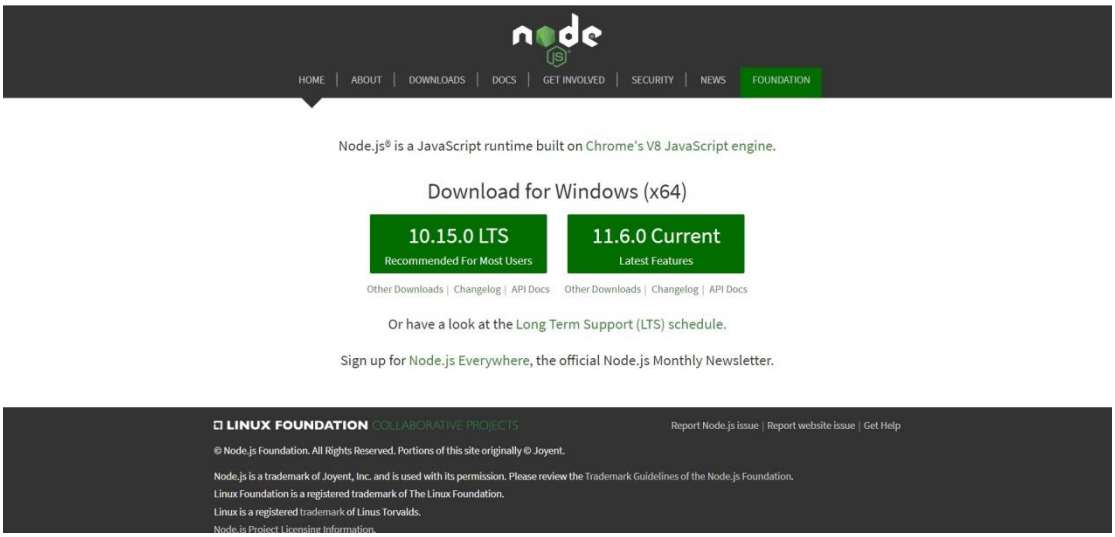
9.1.1 Επιλογή Προγράμματος Υλοποίησης

Για την υλοποίηση της διαδικτυακής εφαρμογής χρειαζόμαστε ένα ολοκληρωμένο περιβάλλον ανάπτυξης λογισμικού(IDE). Υπάρχουν πολλές επιλογές, εμείς επιλέξαμε το Visual Studio Core το οποίο μπορείτε να κατεβάσετε δωρεάν στο https://visualstudio.microsoft.com/. Είναι ένα ισχυρό και έξυπνο IDE που δίνει την καλύτερη κωδικοποίηση για JavaScript, HTML και CSS με αυτόματη συμπλήρωση, χαρακτηριστικά refactoring, διορατικότητα κώδικα αλλά και υποστήριξη δημοφιλών frameworks που ήταν μεταξύ άλλων οι λόγοι οι οποίοι μας ώθησαν να το επιλέξουμε.



Εικόνα 25.Αρχική οθόνη Visual Studio

9.1.2 Εγκατάσταση NodeJS

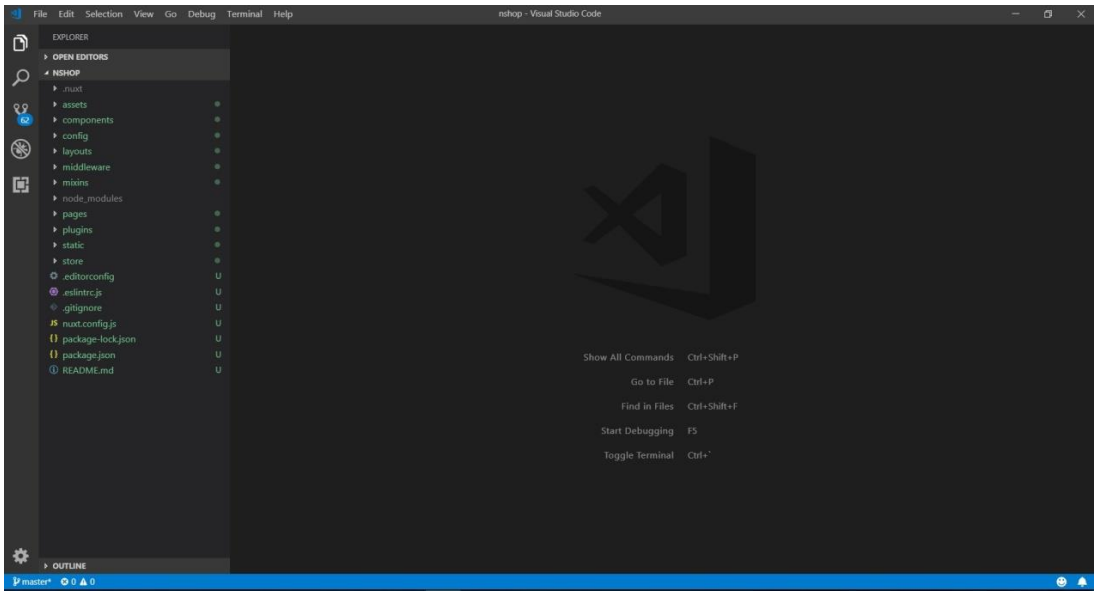


Εικόνα 26.Αρχική οθόνη NodeJS

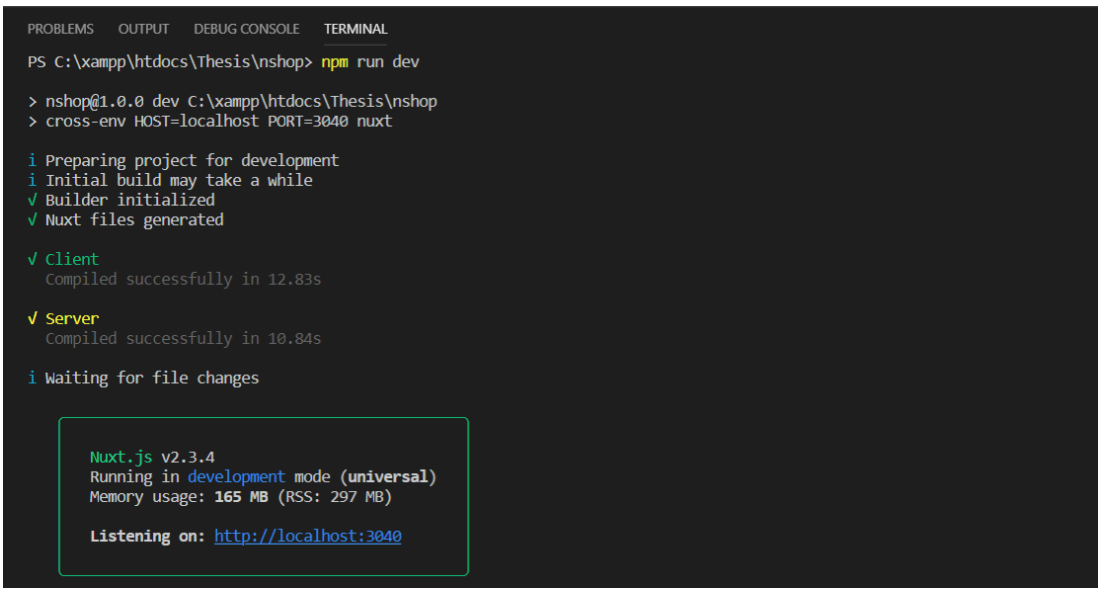
Αφού επιλέξουμε Download v10.15.0 LTS, στα έγγραφα μας εμφανίζεται το αρχείο node-v10.15.0- x64.msi, το οποίο με διπλό κλικ και κάποια επιπλέον βήματα εγκαθίσταται πολύ εύκολα το πρόγραμμα στον υπολογιστή μας.

Μετά την εγκατάσταση του Visual Studio Code και του NodeJS είμαστε έτοιμη για την σχεδίαση της εφαρμογής μας.

Με την εντολή npx create-nuxt-app nshop δημιουργούμε το project μας με όνομα nshop. Το αποτέλεσμα είναι η εικόνα που ακολουθεί.



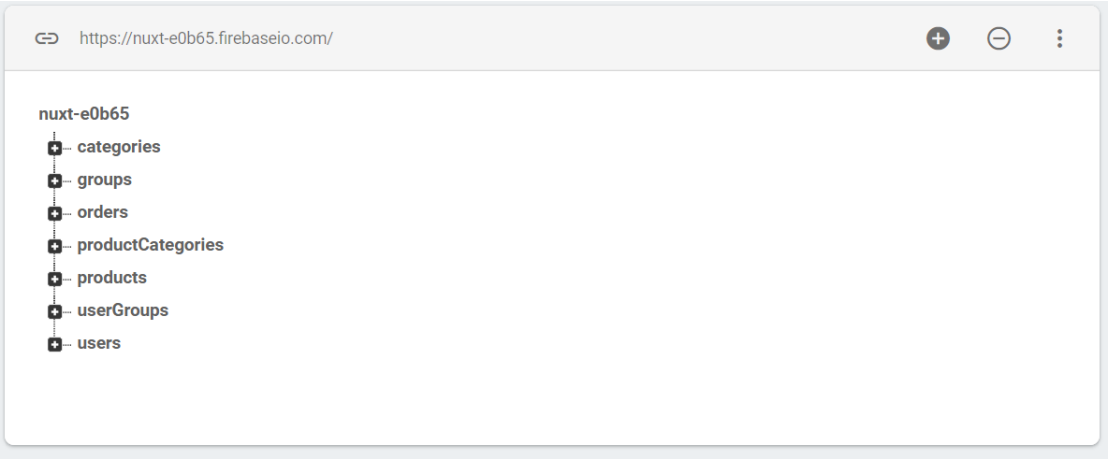
Εικόνα 27.Γραφικό περιβάλλον Visual Studio



Εικόνα 28.Το terminal μετά την εκτέλεση της εντολής npm run dev

9.2 Firebase Data and Rules

Στις παρακάτω εικόνες μπορούμε να δούμε την διάθρωση που έχει η βάση δεδομένων μας και τους κανόνες που χρησιμοποιούμε για να γράψουμε ή να διαβάσουμε δεδομένα σ' αυτήν. Επίσης τον κώδικά που χρησιμοποιήσαμε για να συνδέσουμε την εφαρμογή με την βάση δεδομένων.



Εικόνα 29. Διάρθρωση Firebase

```
{
  "rules": {
    "groups": {
      ".read": "auth != null",
      ".write": "root.child('userGroups').child('-LUzzYgKpblnjgtibpB0').child(auth.uid).exists()",
        ".indexOn": ["name"]
    },
    "userGroups": {
      ".read": "auth != null",
      ".write": "auth != null"
    },
    "users": {
      ".read": "auth != null",
      ".write": "auth != null"
    },
      "categories": {
        ".read": true,
        ".write": "root.child('userGroups').child('-LUzzYgKpblnjgtibpB0').child(auth.uid).exists()"
    },
      "products": {
        ".read": true,
        ".write": "root.child('userGroups').child('-LUzzYgKpblnjgtibpB0').child(auth.uid).exists()",
        ".indexOn": ["name", "status"]
    },
      "productCategories": {
        ".read": true,
        ".write": "root.child('userGroups').child('-LUzzYgKpblnjgtibpB0').child(auth.uid).exists()",
        "$uid": {
          ".indexOn": ["name"]
      }
    },
      "orders": {
        ".read": "auth != null",
        ".write": "auth != null"
    }
  }
}
```

Εικόνα 30.Κανόνες (rules) της Firebase

```
module.exports = {
    fireConfig: {
        apiKey: "AIzfSyDkWAUC1BIDrhuFQwx3j7khWYOHeZCeHVs",
        authDomain: "nuxt-e0b65.firebaseapp.com",
        databaseURL: "https://nuxt-e0b65.firebaseio.com",
        projectId: "nuxt-e0b65",
        storageBucket: "nuxt-e0b65.appspot.com",
        messagingSenderId: "517898105634"
      }
}
```

Εικόνα 31.Ο κώδικας που συνδέει την Ιστοσελίδα με την Firebase

9.3 VeeValidate

Εικόνα 32.Λογότυπο VeeValidate

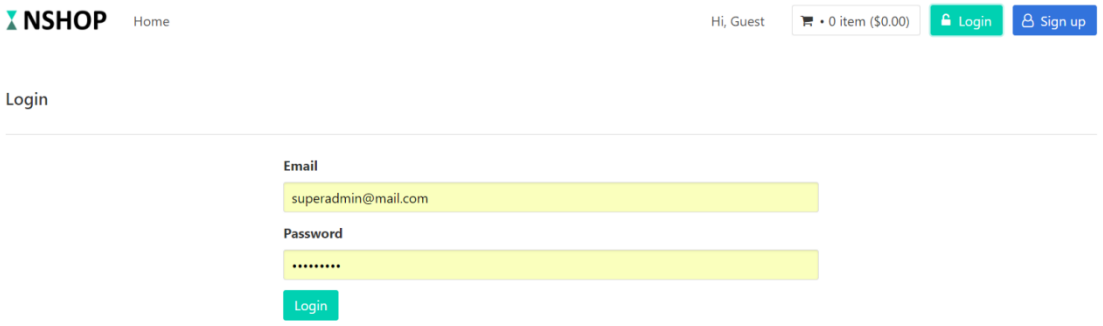
Το VeeValidate είναι μια βιβλιοθήκη επικύρωσης για το Vue.js. Έχει άφθονους κανόνες επικύρωσης και είναι βασισμένο σε πρότυπο, ώστε να είναι παρόμοιο και εξοικειωμένο με το API επικύρωσης της HTML5. Μπορούμε να επικυρώσουμε εισόδους HTML5 καθώς και προσαρμοσμένα στοιχεία Vue.

```
methods: {
      onSubmit () {
        this.$validator.validateAll().then(result => {
          if (result) {
            const productData = {
              name: this.name,
              code: this.code,
              brand: this.brand,
              price: this.price,
              stock: this.stock,
              belongs: this.belongs,
              status: this.status,
              description: this.description,
              image: this.image
            }
```

Εικόνα 33.Παράδειγμα χρήσης VeeValidate

9.4 Λειτουργίες Διαχειριστή (Administrator)

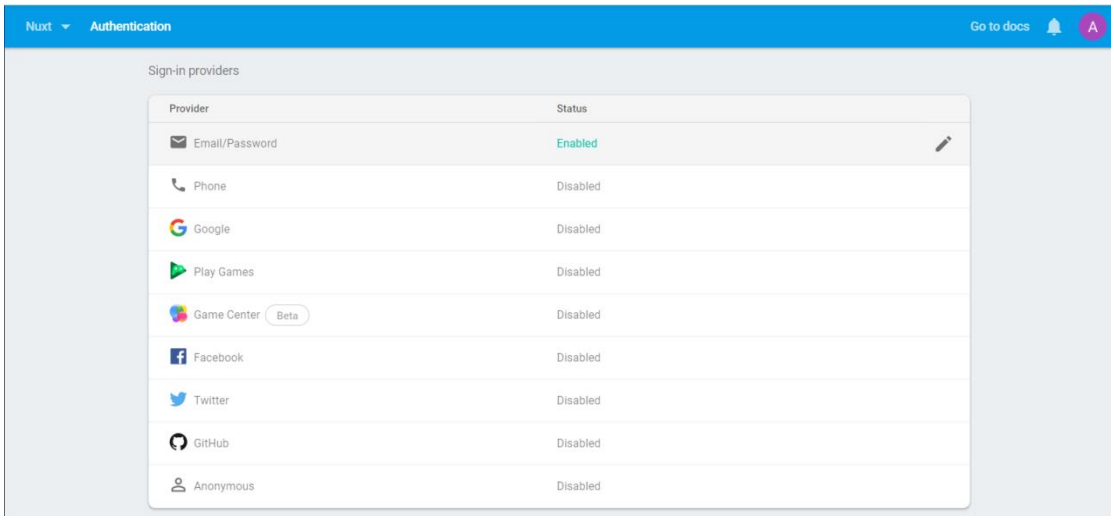
Κάθε διαδικτυακό κατάστημα έχει δύο κατηγορίες χρηστών, τον διαχειριστή και τους πελάτες. Ο Διαχειριστής έχει διαφορετικές λειτουργίες απ' ότι έχει ο απλός πελάτης. Μπορεί να προσθέσει ή να αφαιρέσει προϊόντα, κατηγορίες προϊόντων, να δει τις παραγγελίες που έχουν κάνει οι πελάτες, όπως και να διαχειριστή ομάδες χρηστών.



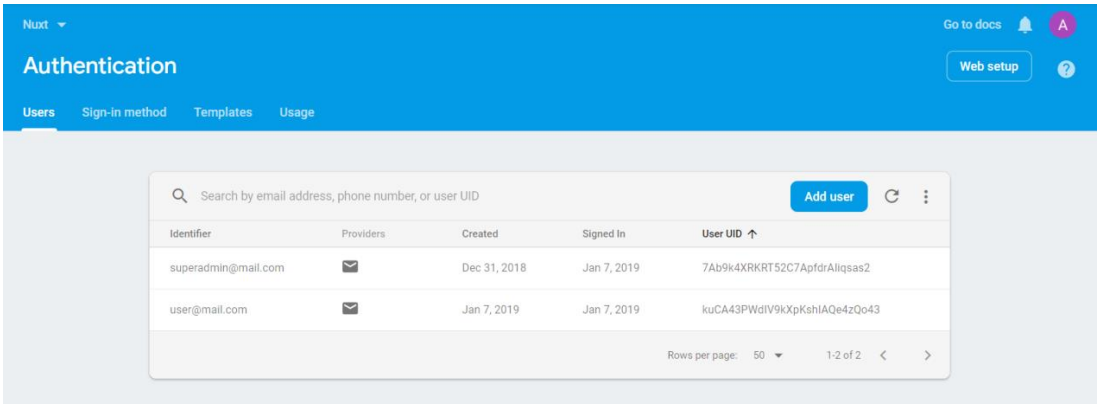
Εικόνα 34.Admin Login

Ο διαχειριστής κάνει Login στο σύστημα μέσω μιας φόρμας πιστοποιήσεις. Η πιστοποίηση των χρηστών (διαχειριστή και πελατών) στη Firebase γίνεται μέσω του e-mail. Σε περίπτωση που κάποιος χρήστης κατά την εγγραφή του (Sign up) χρησιμοποιήσει το ίδιο e-mail με κάποιον άλλον χρήστη που έχει ήδη κάνει εγγραφή στο σύστημα, δεν θα μπορέσει, γιατί δεν θα το επιτρέψει το σύστημα.

Παρακάτω βλέπουμε τις καρτέλες του Authentication της Firebase. Στην πρώτη καρτέλα βλέπουμε τις μεθόδους που μας παρέχει για να κάνει κάποιος Sign-in. Στη δική μας ιστοσελίδα χρησιμοποιήσαμε το Email/Password. Στην δεύτερη καρτέλα βλέπουμε τους χρήστες που έχουν κάνει Sign-in στην εφαρμογή μας.

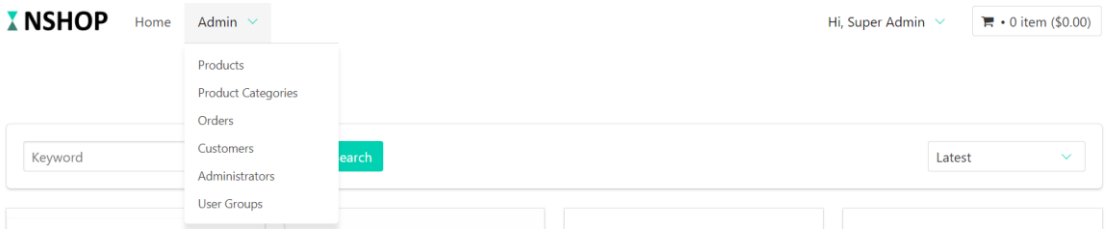


Εικόνα 35.Firebase Authentication



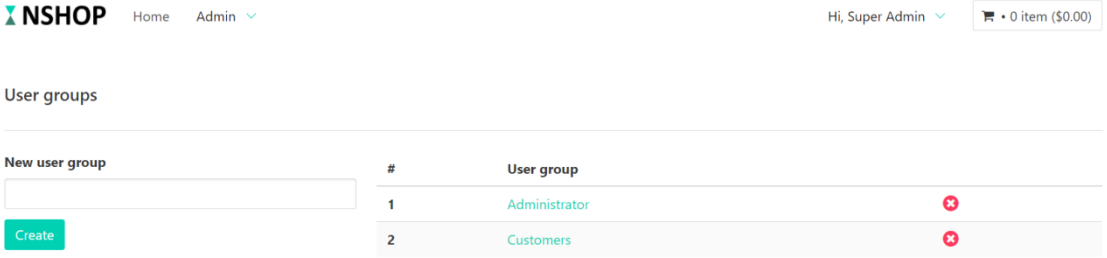
Εικόνα 36.Firebase Authentication Users

Μετά το Login του διαχειριστή δίπλα από το κουμπί Home εμφανίζεται το κουμπί Admin και πατώντας πάνω μια λίστα με τις ενέργειες που έχει ο διαχειριστής και τις οποίες θα αναλύσουμε πιο κάτω.

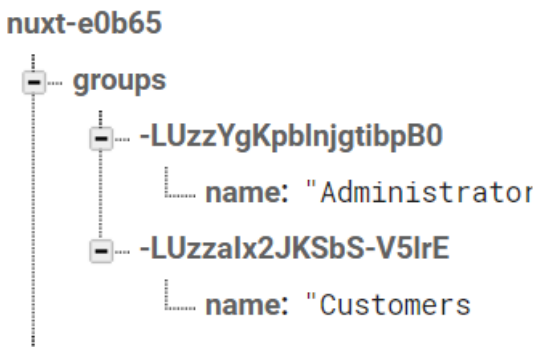


Εικόνα 37.Καρτέλα λειτουργιών Admin

Όπως αναφέραμε ο διαχειριστής μπορεί να δημιουργεί ομάδες χρηστών (User groups). Στις παρακάτω εικόνες βλέπουμε τις ομάδες χρηστών που έχει η εφαρμογή μας και πως είναι διαρθρωμένες στην Firebase. Κάθε εγγραφή του πίνακα (Εικόνα 39) έχει ένα μοναδικό αλφαριθμητικό.

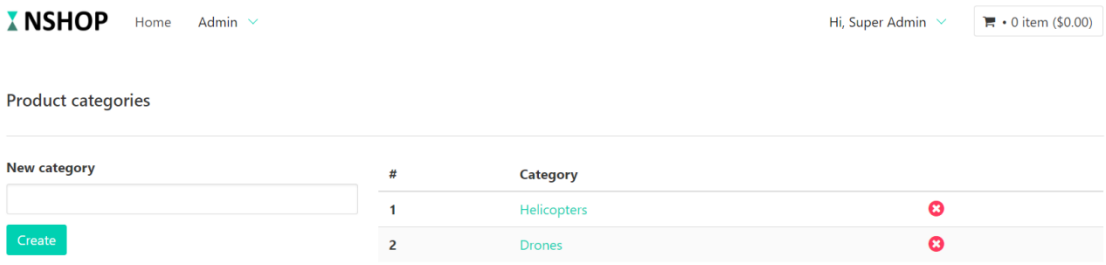


Εικόνα 38.User Groups

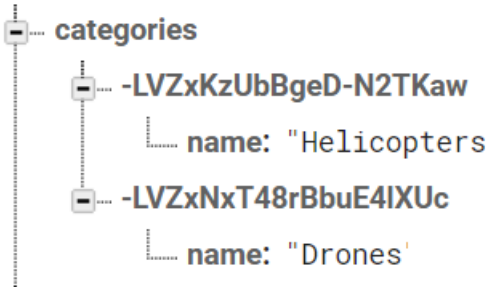


Εικόνα 39.Firebase User Groups

Μια ακόμα λειτουργία που έχει ο διαχειριστής είναι να προσθέτει κατηγορίες προϊόντων. Στις παρακάτω εικόνες βλέπουμε τις κατηγορίες προϊόντων που έχει προσθέσει ο διαχειριστής και την διάθρωση τους στην βάση δεδομένων.



Εικόνα 40.Product categories

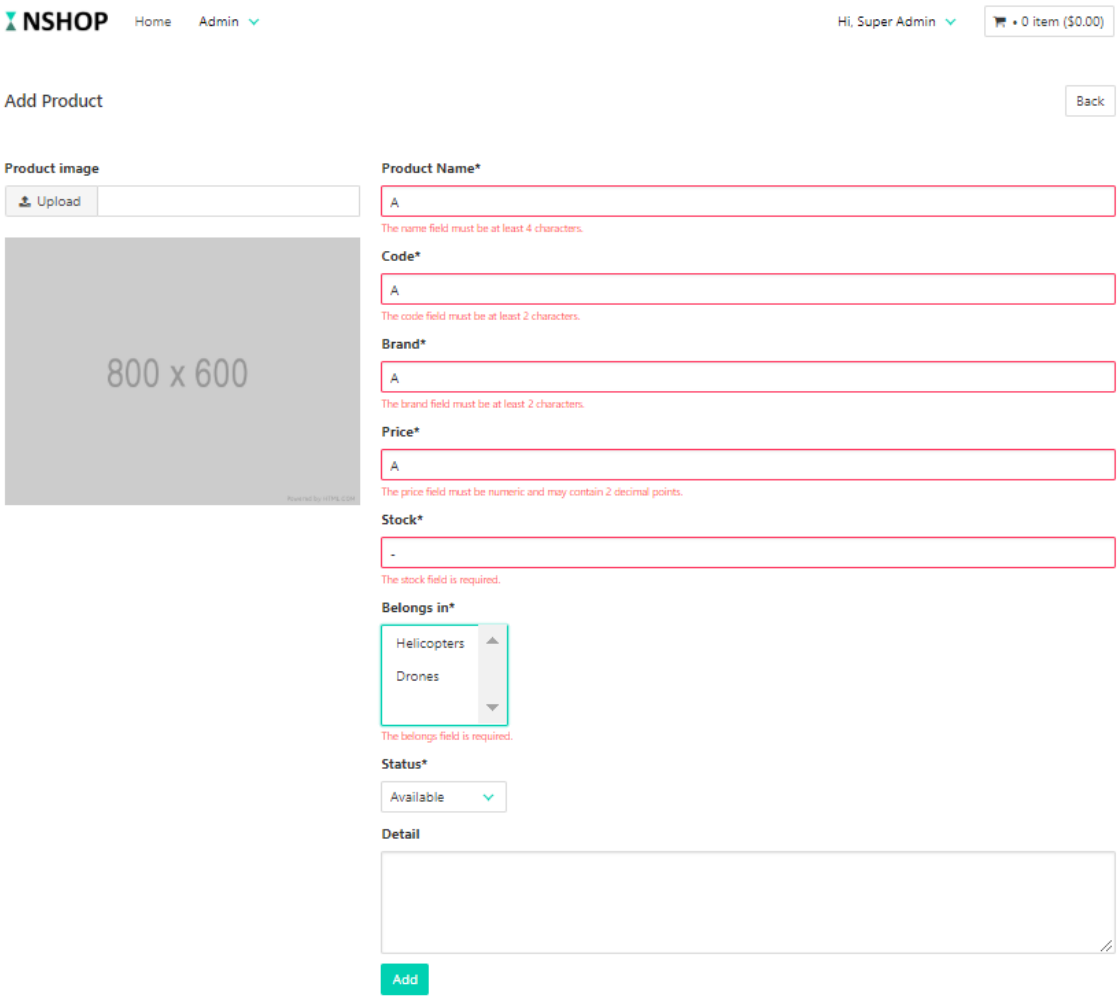


Εικόνα 41.Firebase Product categories

Μια από τις βασικές λειτουργίες του διαχειριστή είναι να προσθέτει προϊόντα στην εφαρμογή. Στην παρακάτω εικόνα μπορούμε να δούμε την καρτέλα με την οποία μπορεί να προσθέσει ή και να τροποποιήσει το προϊόν. Περιλαμβάνει τα πεδία ¨Product Name¨, ¨Code¨, ¨Brand¨, ¨Price¨, ¨Stock¨, ¨Belongs In¨, ¨Status¨ και ¨Detail¨. Σε όλα τα πεδία εφαρμόζεται έλεγχος για την ορθή συμπλήρωση τους μέσω της βιβλιοθήκης vee-validate.

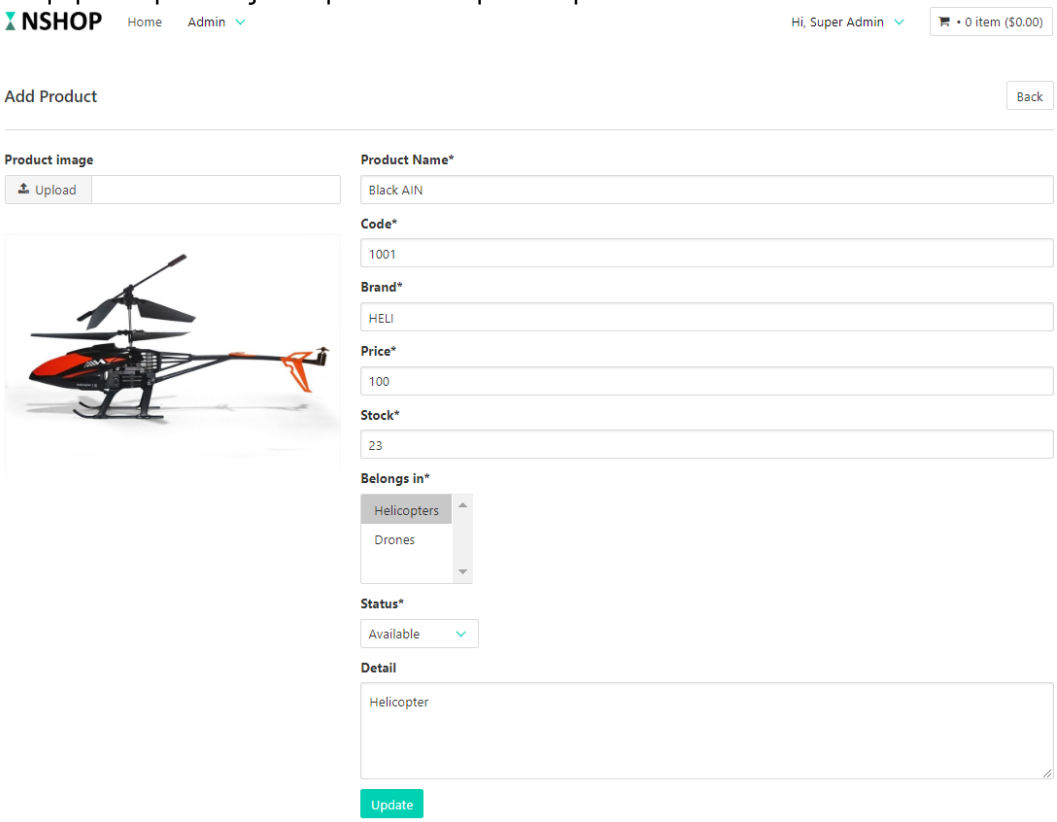
```
<div class="control">
                <input class="input" type="text" name="name" v-model="name" v-
validate="'required|min:4'" :class="{ 'is-danger': errors.has('name') }">
                <p v-show="errors.has('name')" class="help is-danger">{{
errors.first('name') }}</p>
              </div>
```

Στο παραπάνω παράδειγμα κώδικα βλέπουμε τον έλεγχο που πραγματοποιεί το vee-validate στο πεδίο ¨name¨. Αν η καταχώρηση που κάνουμε είναι μικρότερη από τέσσερις χαρακτήρες θα μας εμφανίσει μήνυμα για την σωστή συμπλήρωση του πεδίου. Στην παρακάτω εικόνα βλέπουμε την λειτουργία του vee-validate στα πεδία στη σελίδα Add product.



Εικόνα 42.Add Product

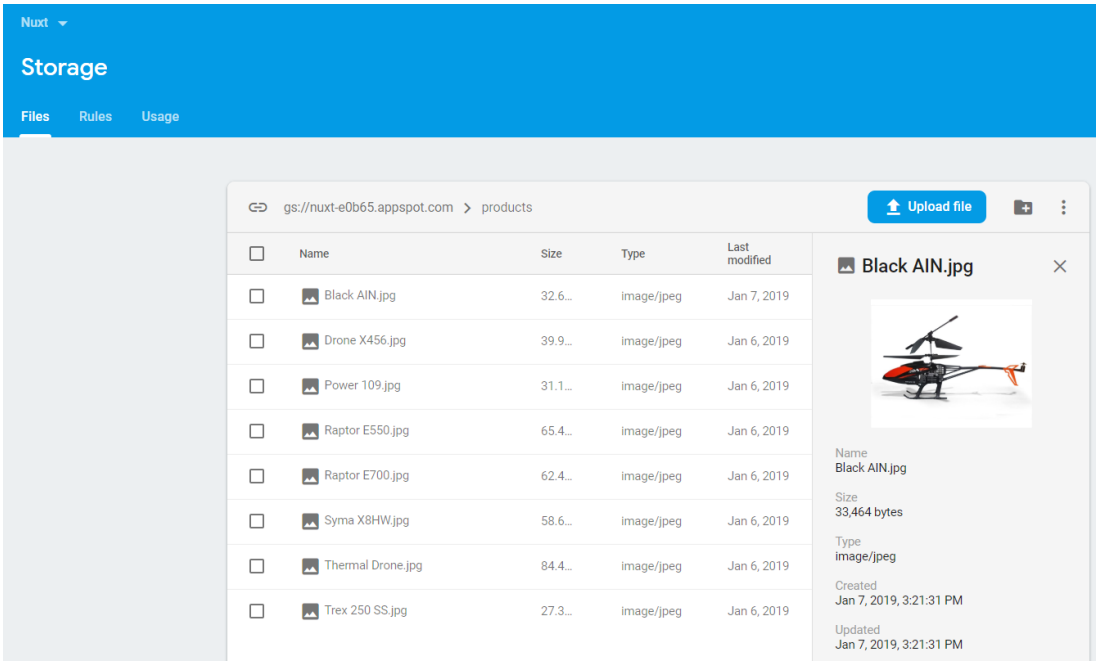
Στην επόμενη εικόνα (εικόνα 43) βλέπουμε συμπληρωμένη την καρτέλα για την προσθήκη του προϊόντος και την απεικόνιση του στην Firebase.



Εικόνα 43.Add Product με συμπληρωμένα τα πεδία

Εικόνα 44.Η εγγραφή του προϊόντος στην Firebase

Επίσης η Firebase μας παρέχει την δυνατότητα μέσω του Storage να αποθηκεύουμε τις εικόνες των προϊόντων μας. Στην εικόνα 44 βλέπουμε την καταχώρηση της φωτογραφίας μας στην βάση και στην εικόνα 45 τον κώδικα που χρησιμοποιήσαμε.

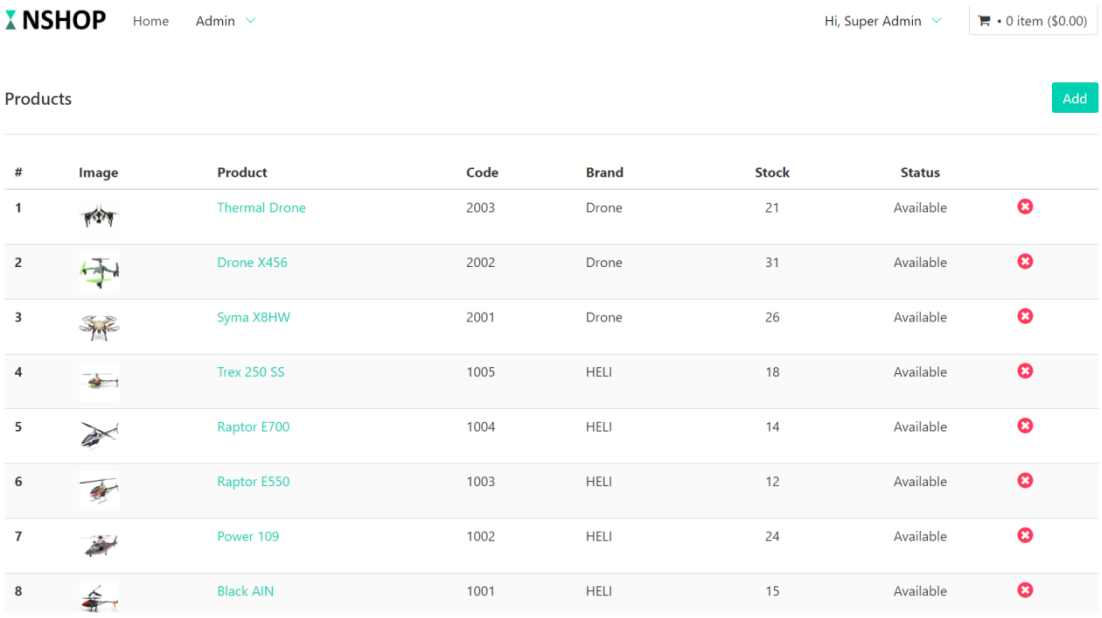


Εικόνα 45.Firebase Image Storage

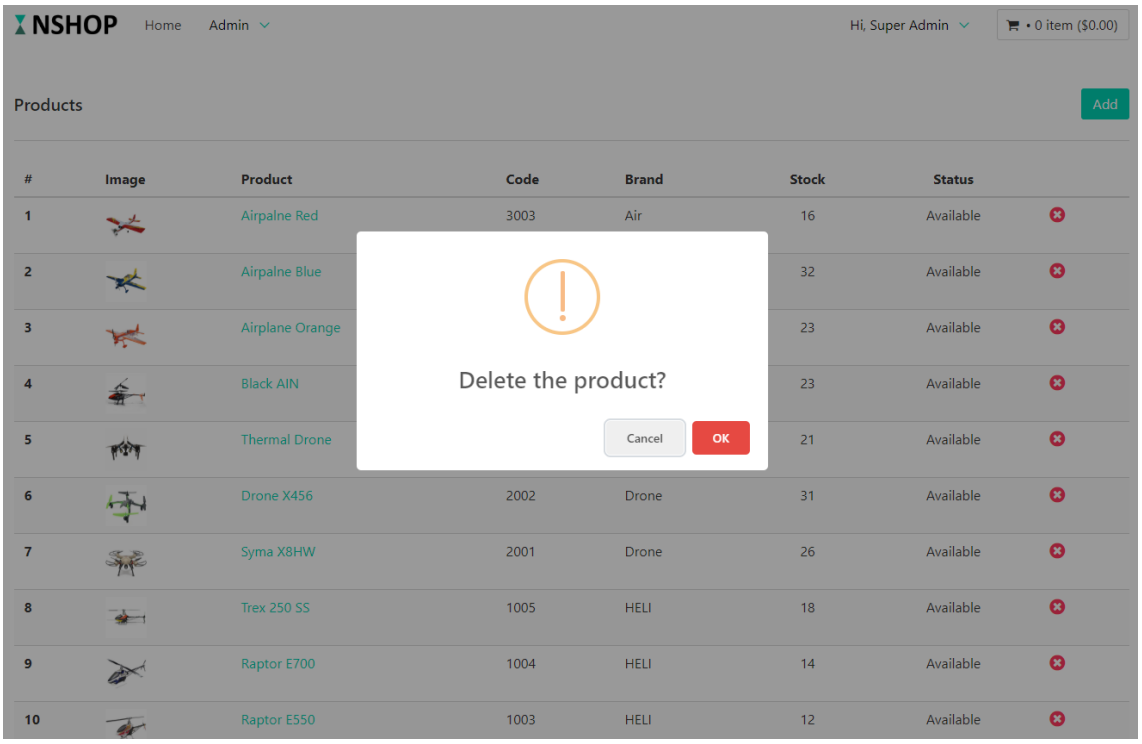
```
service firebase.storage {
  match /b/{bucket}/o {
    match /products {
      match /{allImages=**} {
        allow read;
      }

      match /{allPaths=**} {
        allow write: if request.auth != null;
      }
    }
  }
}
```

Εικόνα 46.Κώδικας για την αποθήκευση των εικόνων στην Firebase

Στην εικόνα 47 μπορούμε να δούμε την σελίδα που έχει τον πλήρη κατάλογο των προϊόντων που έχει προσθέσει ο διαχειριστής και στην οποία ο διαχειριστής μπορεί να προσθέσει, να διαγράψει και να επεξεργαστή το κάθε προϊόν. Επίσης παρέχει στον διαχειριστή την δυνατότητα να δει το απόθεμα των προϊόντων και να τροποποίηση την διαθεσιμότητα.



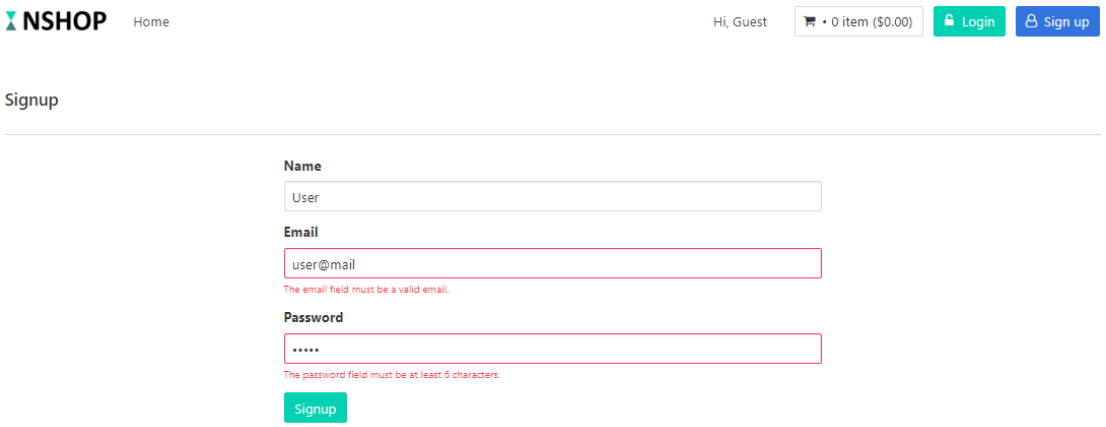
Εικόνα 47.Κατάλογος προϊόντων



Εικόνα 48. Διαγραφή προϊόντος

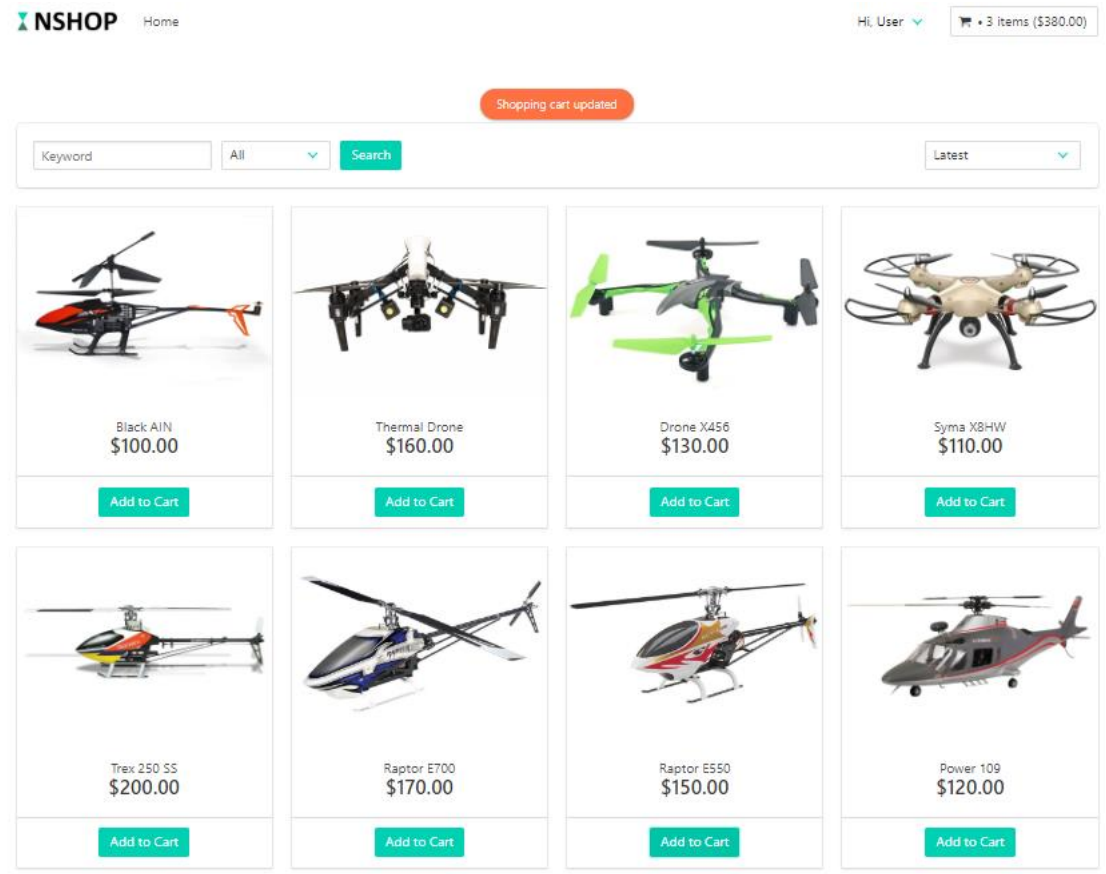
9.5 Λειτουργίες Χρήστη

Ο χρήστης για να παραγγείλει κάποιο προϊόν θα πρέπει να κάνει εγγραφή (Sign up) και να συμπλήρωση σωστά όλα τα πεδία που απαιτούνται, τα οποία είναι ¨Name¨, ¨Email¨ και ¨ Password¨, αλλιώς δεν θα μπορέσει να παραγγείλει. Τον έλεγχο για την σωστή συμπλήρωση των πεδίων το έχει αναλάβει το vee-validate.



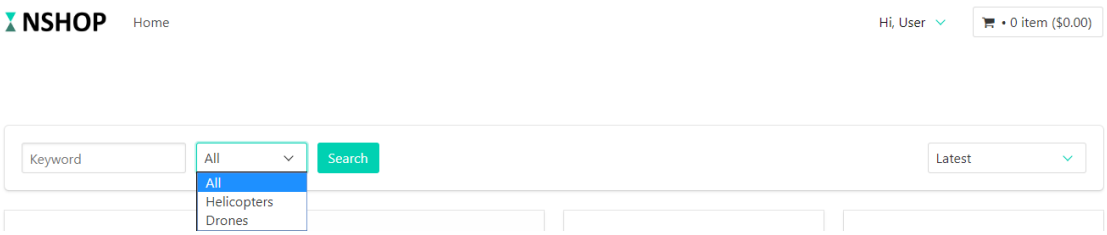
Εικόνα 49.Έλεγχος πεδίων κατά την εγγραφή του χρήστη

Μετά την εγγραφή ο χρήστης μπορεί να αγοράσει κάποιο προϊόν πατώντας το κουμπί ¨Add to Cart¨ και αυτόματα του εμφανίζεται ένα μήνυμα μερικών δευτερολέπτων που λέει ¨Shopping cart upload¨ και πάνω δεξιά εμφανίζεται ο αριθμός των προϊόντων που έχει παραγγείλει και το συνολικό κόστος της παραγγελίας. Παρατηρούμε ότι απουσιάζει η καρτέλα του διαχειριστή δίπλα από του κουμπί ¨Home¨. Αυτό συμβαίνει γιατί ο απλός χρήστης δεν έχει τα δικαιώματα του διαχειριστή.

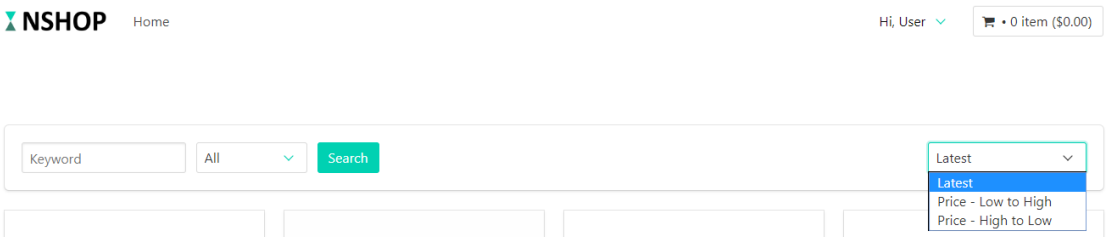


Εικόνα 50.Κατάλογος προϊόντων

Ο χρήστης μπορεί να αναζητήσει το προϊόν βάση της κατηγορίας στην οποία ανήκει ή βάση μιας λέξη κλειδί που θα εισάγει.(εικόνα 51) Επίσης μπορεί να ταξινομήσει τα προϊόντα βάση της τιμής τους ¨χαμηλή->υψηλή¨, υψηλή->χαμηλή (εικόνα 52) και με βάση το τελευταίο προϊόν που προστέθηκε στην εφαρμογή.

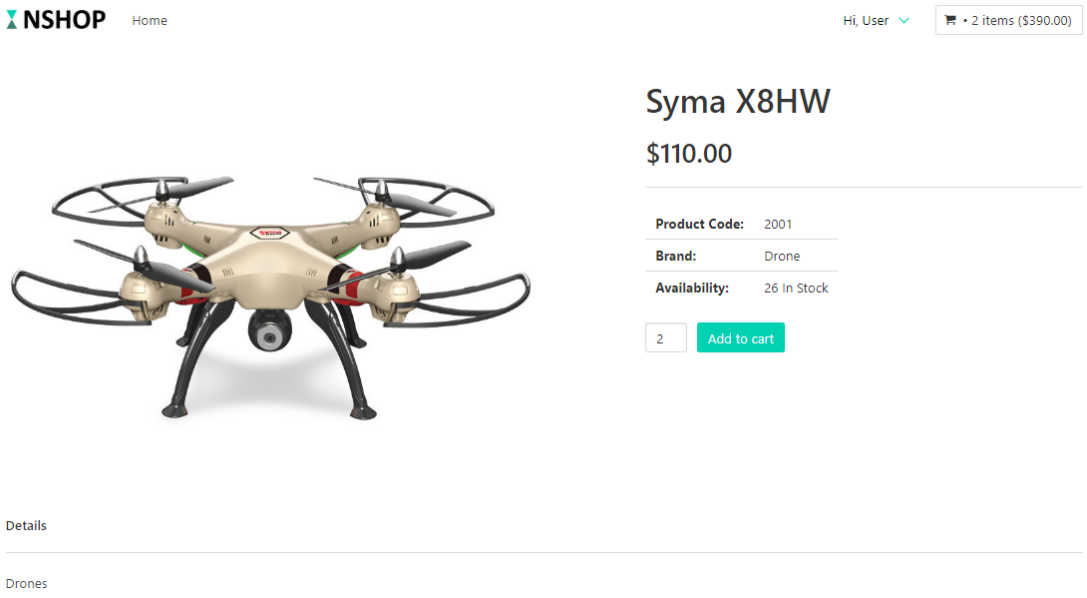


Εικόνα 51.Αναζήτηση βάση κατηγορίας



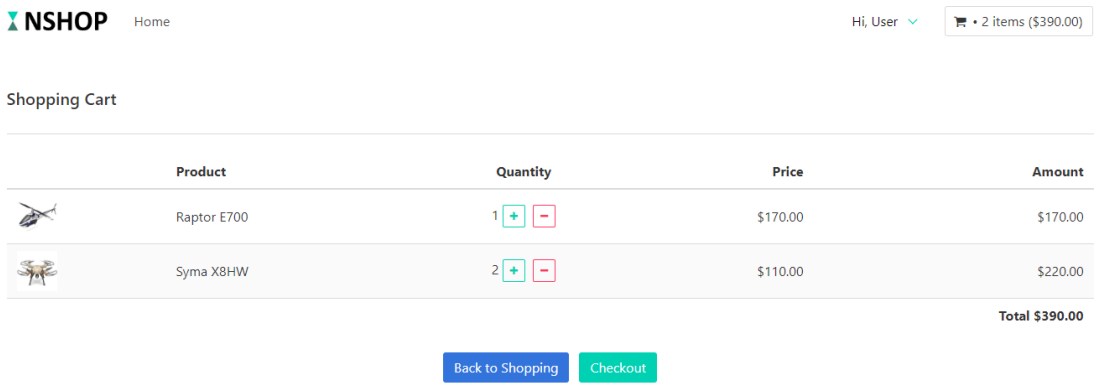
Εικόνα 52.Αναζήτηση βάση τιμής

Επίσης ο χρήστης επιλέγοντας το προϊόν που τον ενδιαφέρει μπορεί να πατήσει πάνω στην εικόνα και να μεταβεί σε άλλη σελίδα που μπορεί να δει αναλυτικά το συγκεκριμένο προϊόν και να το παραγγείλει. Μπορεί να δει τον κωδικό, την μάρκα, την διαθεσιμότητα και την περιγραφή του προϊόντος. Η συγκεκριμένη σελίδα μας δίνει την δυνατότητα να προσδιορίσουμε την ποσότητα που θέλουμε από κάθε προϊόν.



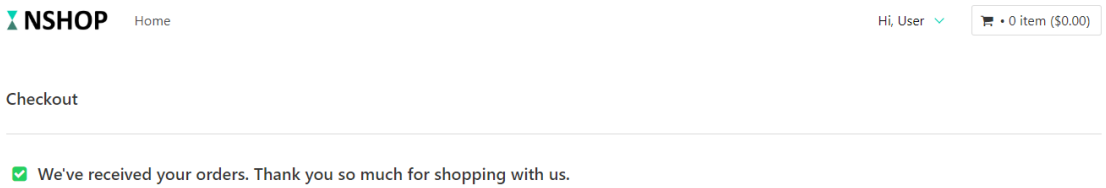
Εικόνα 53.Καρτέλα προϊόντος

Στη συνεχεία ο χρήστης πατώντας το κουμπί πάνω δεξιά με το ¨καροτσάκι¨ μεταβαίνει στην σελίδα (εικόνα 54) για να ολοκληρώσει την παραγγελία του. Σ' αυτή την σελίδα μπορεί να δει αναλυτικά τα προϊόντα που θέλει να παραγγείλει, την ποσότητα αυτών, καθώς και το συνολικό κόστος της παραγγελίας. Επίσης μπορεί να αυξήσει ή να μειώσει την ποσότητα τους.



Εικόνα 54.Shopping Cart

Ο χρήστης πατώντας το κουμπί ¨Checkout¨ ολοκληρώνει την παραγγελία του και του εμφανίζεται το μήνυμα ¨We've received your orders. Thank you so much for shopping with us¨ (εικόνα 55) .Ακολουθεί και η εγγραφή της παραπάνω παραγγελίας στην Firebase (Εικόνα 56).



Εικόνα 55.Checkout

Εικόνα 56.Καταγραφή παραγγελίας στην Firebase

10. Συμπεράσματα και μελλοντικές επεκτάσεις

Το διαδικτυακό κατάστημα που δημιουργήσαμε καλύπτει τις βασικές λειτουργίες που εκτελεί ο διαχειριστής και ο χρήστης χρησιμοποιώντας μια cloud βάση δεδομένων, αντί για μια σχεσιακή βάση δεδομένων. Μπορούμε να δημιουργήσουμε ή να διαγράψουμε χρήστες και να τους δώσουμε διακριτούς ρόλους (διαχειριστή ή απλού χρήστη). Όλοι οι χρήστες πιστοποιούνται από την βάση δεδομένων μας (Firebase) μέσω του mail τους. Ο διαχειριστής μπορεί να εισάγει κατηγορίες προϊόντων στη βάση δεδομένων και να τις επεξεργαστεί/τροποποιήσει. Επίσης μπορεί να εισάγει προϊόντας στις κατηγορίες και να τα επεξεργαστεί/τροποποιήσει. Ο χρήστης μπορεί να κάνει αναζήτηση του προϊόντος ανάλογα με το όνομα του ή την κατηγορία στην οποία ανήκει. Όπως και ταξινόμηση βάση της τιμής του. Επίσης μπορεί να καταχωρήσει με επιτυχία μια παραγγελία και αυτή να εισαχθεί στη βάση δεδομένων.

Μελλοντικά, μπορούμε να εισάγουμε ένα σύστημα πληρωμών, όπως είναι το paypal, ώστε να παρέχεται στους χρήστες η δυνατότητα μετά την ολοκλήρωση της παραγγελίας τους και η πληρωμή της, η οποία θα καταγράφεται στη βάση δεδομένων μας. Μ’ αυτόν τον τρόπο ο διαχειριστής θα γνωρίζει αν κάποιος χρήστης έχει πληρώσει την παραγγελία του ή όχι. Επιπλέον μπορούμε να κρατάμε το ιστορικό από τα προϊόντα με τις μεγαλύτερες πωλήσεις, ώστε μ’ αυτόν τον τρόπο να διαχειριζόμαστε καλύτερα τα αποθέματα των προϊόντων μας, για την καλύτερη εξυπηρέτηση των πελατών μας.

11. Βιβλιογραφία

1. https://en.wikipedia.org/wiki/HTML
2. https://en.wikipedia.org/wiki/Cascading_Style_Sheets
3. https://en.wikipedia.org/wiki/JavaScript
4. https://vuejs.org/
5. https://nuxtjs.org/
6. https://nodejs.org/en/
7. https://firebase.google.com/
8. https://en.wikipedia.org/wiki/Node.js
9. https://en.wikipedia.org/wiki/NoSQL
10. https://en.wikipedia.org/wiki/JavaScript_framework
11. https://en.wikipedia.org/wiki/Unified_Modeling_Language
12. https://baianat.github.io/vee-validate/
13. https://vuex.vuejs.org/
14. https://www.npmjs.com/package/slugify
15. https://www.npmjs.com/package/vue-swal
16. https://www.npmjs.com/package/vue-warehouse
17. https://en.wikipedia.org/wiki/Web_service