ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«ΑΝΑΠΤΥΞΗ ΕΝΟΣ ΣΥΣΤΗΜΑΤΟΣ ΔΙΑΧΕΙΡΙΣΗΣ ΠΕΡΙΕΧΟΜΕΝΟΥ (CMS) ΒΑΣΙΣΜΕΝΟ ΣΕ NoSQL ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ»

Της φοιτήτριας
Παλαιοχωρινού Ευαγγελία
Αρ. Μητρώου: 113768

Επιβλέπων
Ονοματεπώνυμο Αντωνίου
Ευστάθιος
Βαθμίδα ……

Ημερομηνία Σεπτέμβριος 2020

Τίτλος Δ.Ε. Ανάπτυξη ενός Συστήματος Διαχείρισης Περιεχομένου (CMS) βασισμένο σε NoSQL βάση δεδομένων

Κωδικός Δ.Ε. …

Ονοματεπώνυμο φοιτητή/τών Ευαγγελία Παλαιοχωρινού

Ονοματεπώνυμο εισηγητή Ευστάθιος Αντωνίου

Ημερομηνία ανάληψης Δ.Ε. …

Ημερομηνία περάτωσης Δ.Ε. …

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

Πρόλογος

Η εισαγωγή των Μη Σχεσιακών βάσεων δεδομένων στον τομέα της ανάπτυξης εφαρμογών (διαδικτυακών και μη) μπορεί να προσφέρει πολύ καλύτερη εμπειρία χρήσης με πολύ χαμηλότερο κόστος και προσπάθεια από μεριάς προγραμματισμού.

Τα δεδομένα του διαδικτύου, στις μέρες μας γίνονται ολοένα και πιο μεγάλα σε όγκο και γίνεται πιο δύσκολο έως ακατόρθωτο να ομαδοποιηθούν. Επομένως, για τον όλο και μεγαλύτερο όγκο δεδομένων, οι Βάσεις Δεδομένων των εφαρμογών που τα διαχειρίζονται θα πρέπει συνεχώς να προσαρμόζονται στις απαιτήσεις των δεδομένων αυτών.

Στο διαδίκτυο συνεχώς αυξάνονται οι τύποι των δεδομένων που πρέπει να αποθηκεύονται σε μια βάση δεδομένων (φωτογραφίες, βίντεο, ετικέτες, δημοσιεύσεις παντός τύπου κλπ). Φαινόμενα σαν και αυτό κάνουν τις Σχεσιακές Βάσεις Δεδομένων να μην μπορούν να ανταποκριθούν, αφού κάθε νέος τύπος αρχείων που εισάγεται απαιτεί να επανασχεδιάζεται σχεδόν από την αρχή μια Σχεσιακή Βάση Δεδομένων. Κάτι τέτοιο δεν απαιτείται όταν χρησιμοποιηθεί μια Μη Σχεσιακή Βάση Δεδομένων, που από τον σχεδιασμό της είναι ικανή να αποθηκεύει οποιονδήποτε τύπο δεδομένων χωρίς περιορισμούς και ανεξάρτητα από το πόσο περίπλοκος είναι.

Η παρούσα πτυχιακή εργασία επιλέχθηκε για να δειχθεί το πόσο σημαντική είναι η μετάβαση στις μη σχεσιακές βάσεις δεδομένων, επιλέγοντας την MongoDB ως βάση δεδομένων για την ανάπτυξη ενός Συστήματος Διαχείρισης Περιεχομένου (CMS).

Περίληψη

Στόχος της παρούσας πτυχιακής εργασίας είναι η ανάπτυξη ενός συστήματος διαχείρισης περιεχομένου (CMS) με τη χρήση NoSQL βάσης δεδομένων. Από όλες τις NoSQL βάσεις δεδομένων, επιλέχθηκε η MongoDB η οποία ανήκει στην κατηγορία των Document Stores. Ο λόγος που επιλέχθηκε η MongoDB, είναι γιατί πρόκειται για μία βάση δεδομένων η οποία παρέχει μεγάλη ευελιξία στον προγραμματιστή, υψηλή ταχύτητας εγγραφή και μεγάλη αποδοτικότητα σε queries.

Για το παραδοτέο κομμάτι της εργασίας αυτής, έγινε χρήση της Express js, της Nodejs και της MongoDB.

Στην πλατφόρμα που αναπτύχθηκε, υπάρχουν 3 ρόλοι χρηστών. Οι επισκέπτες, οι συγγραφείς και οι διαχειριστές.

Οι επισκέπτες έχουν τη δυνατότητα να δουν τα άρθρα που έχουν δημοσιευτεί και να αφήσουν σχόλια σε αυτά.

Οι χρήστες που έχουν εγγραφεί και έχουν κάνει είσοδο με τα στοιχεία σύνδεσής τους, δηλαδή οι συγγραφείς, μπορούν να δημιουργήσουν νέα άρθρα ή να επεξεργαστούν ήδη υπάρχοντα. Επιπρόσθετα, τα άρθρα συγκαταλέγονται σε κατηγορίες. Οι συγγραφείς μπορούν να κατηγοριοποιήσουν το κάθε άρθρο και να δημιουργήσουν νέες κατηγορίες.

Οι διαχειριστές από την πλευρά τους μπορούν να εγκρίνουν τα σχόλια των επισκεπτών και έχουν όσες δυνατότητες έχουν και οι συγγραφείς, ενώ μπορούν επίσης να διαγράψουν άρθρα και κατηγορίες.

«Development of a Content Management System, using a NoSQL database»

«Evaggelia Palaiochorinou»

Abstract

The goal of this thesis is to develop a content management system (CMS) using a NoSQL database. Among all NoSQL databases, we selected MongoDB, which is included on the Document Stores category. The reason why MongoDB was selected, is because it is a database which provides flexibility to the programmer, quick write operations and query efficiency.

For the project of this thesis, we used Express js, Nodejs and MongoDB. The CMS platform that was developed, illustrates 3 user roles. The guests, the authors and the administrators.

The guests are able to see the uploaded posts and make comments. The users that are registered and have logged in with their credentials, the authors, can create new posts or edit existing posts. The posts can be included in different categories. The authors can categorize each post and create new categories. The administrators can approve the guests' comments and have all the authors' privileges, plus the posts' and categories deletion.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά την οικογένειά μου για την ηθική υποστήριξη, όπως επίσης και τους συναδέλφους μου, για την καθοδήγηση και την κατανόηση που έδειξαν καθ' όλη τη διάρκεια της πτυχιακής μου εργασίας.
Επίσης θα ήθελα να πω ένα μεγάλο ευχαριστώ στον κύριο Ευστάθιο Αντωνίου, τον επιβλέποντα καθηγητή της παρούσας εργασίας, για την άψογη συνεργασία και καθοδήγηση που μου παρείχε.

Περιεχόμενα

Πρόλογος....iv
Περίληψη....v
Abstract....vi
Ευχαριστίες....vii
Περιεχόμενα....viii
Κατάλογος Εικόνων....xii
Κατάλογος Πινάκων....xii
Κεφάλαιο 1ο: Μη Σχεσιακές Βάσεις Δεδομένων....1
1.1 Εισαγωγή....1
1.2 Το θεώρημα CAP....1
1.3 Χαρακτηριστικά Μη Σχεσιακών Βάσεων Δεδομένων....2
1.4 Πλεονεκτήματα Μη Σχεσιακών Βάσεων Δεδομένων....2
1.5 Μειονεκτήματα Μη Σχεσιακών Βάσεων Δεδομένων....2
1.6 Κατηγορίες NoSQL Συστημάτων....3
1.6.1 Key Value Stores....3
1.6.2 Document Stores....4
1.6.3 Column Stores....4
1.6.4 Βάσεις Δεδομένων Γράφων....5
1.7 Επίλογος....5
Κεφάλαιο 2ο: MongoDB....6
2.1 Εισαγωγή....6
2.2 Έγγραφα....7
2.2.1 Δομή των Εγγράφων....7
2.2.2 Ονόματα Πεδίων....8
2.2.3 Πλεονεκτήματα των εγγράφων....8
2.3 Χαρακτηριστικά της MongoDB....8
2.4 Εγκατάσταση της MongoDB σε Windows:....9
2.4.1 MongoDB Help....10
2.4.2 MongoDB Statistics....11
2.4.3 Mongo shell (Κέλυφος)....11
2.5 Συλλογές (Collections)....12
2.6 Βασικές Λειτουργίες (CRUD Operations)....12

2.6.1 Δημιουργία (Insert) 13
2.6.2 Αναζήτηση (Query) 14
2.6.3 Ενημέρωση (Update) 15
2.6.4 Διαγραφή (Delete) 18
2.7 Aggregation (Ομαδοποίηση) 19
2.8 MongoDB Replication 20
2.8.1 Πώς γίνεται το Replication στη MongoDB 20
2.8.2 Δημιουργία ενός replica set 21
2.8.3 Προσθήκη κόμβων σε ένα replica set 21
2.8.4 Πλεονεκτήματα χρήσης της μεθόδου replication 22
2.8.5 Χαρακτηριστικά των replica sets 22
2.9 MongoDB Sharding 22
2.9.1 Πλεονεκτήματα χρήσης της μεθόδου sharding 23
2.10 Πότε δεν συνιστάται η χρήση της MongoDB 23
2.11 Θέματα ασφαλείας στη MongoDB 23
2.11.1 Αρχεία δεδομένων της MongoDB 23
2.11.2 Διεπαφές χρηστών (Client Interfaces): 24
2.11.3 Πιθανότητες για μολυσματικές επιθέσεις 24
2.11.4 Αυθεντικοποίηση (Authentication) 25
2.11.5 Εξουσιοδότηση (Authorization) 25
2.11.6 Auditing (Έλεγχος) 26
2.12 Επίλογος 26
Κεφάλαιο 3ο: CMS (Content Management System) 27
3.1 Εισαγωγή 27
3.2 Ορισμός- Περιγραφή 28
3.3 Πλεονεκτήματα 29
3.3.1 Γενικά Πλεονεκτήματα 29
3.3.2 Ειδικά Πλεονεκτήματα 30
3.4 Χαρακτηριστικά 32
3.5 Βασικά Χαρακτηριστικά 32
3.6 Είδη Συστημάτων Διαχείρισης Περιεχομένου 34
3.6.1 ASP και Licensed (με βάση το χώρο αποθήκευσης και διαχείρισης) 34
3.6.2 Commercial, Open source, Managed Open Source (με βάση το είδος του παρόχου) 35
3.7 Κριτήρια Επιλογής CMS 35
3.8 Επίλογος 37

Κεφάλαιο 4ο: Τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη του CMS με MongoDB 38
4.1 Εισαγωγή 38
4.2 Node.js 38
4.3 Express 38
Κεφάλαιο 5ο: Βασικές Λειτουργίες του CMS που αναπτύχθηκε και πώς υλοποιήθηκαν 40
5.1 Εισαγωγή 40
5.2 Ρόλοι Χρηστών 40
5.3 Δομή του Project 40
5.4 Σχήμα Βάσης Δεδομένων 42
5.5 Σύνδεση στη βάση δεδομένων 43
5.6 Εκκίνηση του web server 44
5.7 Index page 44
5.8 Read More 44
5.9 Comment 45
5.10 Approve Comments 45
5.11 Register 46
5.12 Login 47
5.13 Create New Post 48
5.14 Edit Post 49
5.15 Delete Post 50
5.16 Categories 50
5.17 New Category 50
5.18 Edit Category 51
5.19 Delete Category 51
5.20 Logout 51
5.21 Επίλογος 52
Κεφάλαιο 6ο: Οδηγίες εκτέλεσης της εφαρμογής τοπικά 53
6.1 Εισαγωγή 53
6.2 Εκκίνηση του MongoDB Server 53
6.3 Project Build & Εκκίνηση web server 53
6.4 Επίλογος 53
Κεφάλαιο 7ο: Οδηγός Χρήσης 54
7.1 Για τον Επισκέπτη 54
7.1.1 Προβολή Άρθρων 54
7.1.2 Σχολιασμός 54

7.2 Για τον Συγγραφέα 54
7.2.1 Register 54
7.2.2 Login 55
7.2.3 Author Dashboard 55
7.2.4 Posts 56
7.2.5 Categories 57
7.2.6 LogOut 57
7.3 Για τον Διαχειριστή 57
7.3.1 Admin Login 57
7.3.2 Admin Dashboard 57
7.3.3 Posts 58
7.3.4 All Posts 58
7.3.5 Edit Post 58
7.3.6 Create Posts 58
7.3.7 Delete Posts 59
7.3.8 Comments 59
7.3.9 Categories 59
7.3.10 Logout 59
7.4 Επίλογος 60
Κεφάλαιο 8ο: Συμπεράσματα & Περιθώρια Βελτίωσης 61
8.1 Συμπεράσματα 61
8.2 Περιθώρια Βελτίωσης 61
ΒΙΒΛΙΟΓΡΑΦΙΑ 62

Κατάλογος Εικόνων

Εικόνα 1. Θεώρημα CAP 1
Εικόνα 2. Διαφορετικού τύπου δεδομένα τα οποία συσχετίζονται με το αντίστοιχο κλειδί 3
Εικόνα 3. Document Stores 4
Εικόνα 4. Column Stores 5
Εικόνα 5. Παράδειγμα ενός document 6
Εικόνα 6. Mongo Help 11
Εικόνα 7. MongoDB Statistics 11
Εικόνα 8. MongoDB Replication 21
Εικόνα 9. MongoDB Sharding 22
Εικόνα 10. Η βάση της λειτουργίας όλων των CMS 29
Εικόνα 11. Τυπικό διάγραμμα ροής εργασίας σε ένα CMS 33
Εικόνα 12. Κεντρική Σελίδα 54
Εικόνα 13. Read More & Comments 54
Εικόνα 14. Register form 55
Εικόνα 15. Login form 55
Εικόνα 16. Author Dashboard 55
Εικόνα 17. Author Dashboard – All Posts 56
Εικόνα 18. Author Dashboard – Edit Post 56
Εικόνα 19. Author Dashboard – Create New Post 56
Εικόνα 20. Author Dashboard - Categories 57
Εικόνα 21. Admin Login 57
Εικόνα 22. Admin Dashboard 58
Εικόνα 23. Admin Dashboard – All Posts 58
Εικόνα 24. Admin Dashboard – Edit Post 58
Εικόνα 25. Admin Dashboard – Create New Post 59
Εικόνα 26. Admin Dashboard - Comments 59
Εικόνα 27. Admin Dashboard - Categories 59

Κατάλογος Πινάκων

Πίνακας 1. Βασικές ιδιότητες των διαφόρων μοντέλων 5
Πίνακας 2. Insert μέθοδοι της MongoDB 14
Πίνακας 3. Update μέθοδοι της MongoDB 17
Πίνακας 4. Delete μέθοδοι της MongoDB 18

Κεφάλαιο 1ο: Μη Σχεσιακές Βάσεις Δεδομένων

1.1 Εισαγωγή

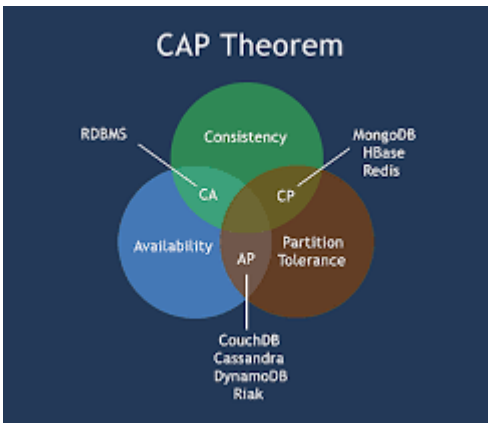
Για αρκετά χρόνια, οι περισσότεροι οργανισμοί και οι επιχειρήσεις, βασιζόταν στη χρήση συστημάτων διαχείρισης σχεσιακών βάσεων δεδομένων (RDBMSs) για την αποθήκευση, επεξεργασία και ανάλυση πληροφοριών. Όταν όμως έφτασε η εποχή των Big Data, χρειάστηκε να αναπτυχθεί ένα νέο είδος βάσεων δεδομένων. Το κυριότερο κίνητρο για την ανάπτυξη των NoSQL, ήταν η επεξεργασία των μεγάλων όγκων δεδομένων που έφερε το Internet. Η ιδέα των NoSQL υπήρχε ήδη για αρκετές δεκαετίες, καθώς πολλές επιχειρήσεις τη δεκαετία του ’60, χρησιμοποιούσαν μη σχεσιακές βάσεις δεδομένων για να αποθηκεύσουν και να εξάγουν μη δομημένη πληροφορία.

Η NoSQL αναπτύχθηκε αρχικά ως λύση για τη διαχείριση του μεγάλου όγκου δεδομένων και την ανάγκη για ταχύτερη επεξεργασία μη δομημένων πληροφοριών. Το μοντέλο NoSQL χρησιμοποιεί κατανεμημένο σύστημα βάσεων δεδομένων, δηλαδή ένα σύστημα που αποτελείται από πολλούς διαφορετικούς υπολογιστές. Τα μη σχεσιακά συστήματα είναι γρηγορότερα, χρησιμοποιούν μία ad-hoc μέθοδο για την οργάνωση των δεδομένων και μπορούν να επεξεργαστούν μεγάλο όγκο από διαφορετικούς τύπους δεδομένων. Όπως έχει αποδειχθεί από έρευνες, η χρήση NoSQL βάσεων δεδομένων είναι η καλύτερη όταν πρόκειται για διαχείριση μεγάλου όγκου μη δομημένων πληροφοριών, όπως επίσης και για γρηγορότερη διαχείριση Big Data. Το γεγονός αυτό οδήγησε οργανισμούς οι οποίοι διαχειρίζονται Big Data, όπως το Facebook, το Twitter, το LinkedIn και το Google, να υιοθετήσουν τη χρήση NoSQL συστημάτων.

1.2 Το θεώρημα CAP

Το θεώρημα CAP (Consistency, Availability, Partition tolerance) (Συνέπεια, Διαθεσιμότητα, ανοχή Διαμερισμού) [1], γνωστό επίσης και ως θεώρημα Brewer, είναι ένα σημαντικό κομμάτι των μη σχεσιακών βάσεων δεδομένων. Σύμφωνα με το θεώρημα αυτό, είναι αδύνατο για ένα κατανεμημένο σύστημα υπολογιστών να παρέχει ταυτόχρονα περισσότερες από δύο από τις τρεις παρακάτω εγγυήσεις:

- **Συνέπεια:** Εγγύηση ότι όλοι οι κόμβοι βλέπουν τα ίδια δεδομένα την ίδια στιγμή και κάθε συναλλαγή οδηγεί τη βάση δεδομένων από τη μία συνεπή κατάσταση στην άλλη.
- **Διαθεσιμότητα:** Αναφέρεται στην ικανότητα το συστήματος να είναι πάντα διαθέσιμο.
- **Ανοχή διαμερισμού:** Εγγύηση ότι το σύστημα συνεχίζει να λειτουργεί παρά τον διαμερισμό που μπορεί να προκληθεί από βλάβες του δικτύου. Συνήθως αναφέρεται σε κόμβους οι οποίοι είναι γεωγραφικά διαχωρισμένοι και δεν είναι δυνατή η απευθείας επικοινωνία μεταξύ τους.



Εικόνα 1. Θεώρημα CAP

Σε ένα κατανεμημένο περιβάλλον, είναι πολύ πιθανό να υπάρχουν σφάλματα στο δίκτυο, τα οποία μπορεί να προκαλέσουν διαμερισμό στο σύστημα, που σημαίνει ότι μπορεί να χαθούν πακέτα δεδομένων ή κάποιοι από στους κόμβους μπορεί να τεθούν εκτός λειτουργίας. Τα περισσότερα κατανεμημένα συστήματα θα πρέπει να είναι ανεκτικά σε τέτοιου είδους θέματα. Από την άλλη όμως, υπάρχουν συστήματα τα οποία μπορεί να μην είναι ανεκτικά σε διαμερισμό, αλλά μπορούν να είναι συνεπείς και διαθέσιμα οποιαδήποτε στιγμή. Υπάρχουν, επίσης, συστήματα τα οποία δε διαθέτουν συνέπεια, αλλά πρέπει να είναι ανεκτικά σε διαμερισμούς, όπως για παράδειγμα το Facebook.

Συνεπώς, σε περίπτωση βλάβης στο δίκτυο, θα πρέπει να αποφασιστεί είτε η ακύρωση της εργασίας με επίπτωση στη διαθεσιμότητα αλλά εγγύηση της συνέπειας, είτε η συνέχιση της εργασίας με εγγύηση της διαθεσιμότητας, αλλά επίπτωση στην συνέπεια. [2]

1.3 Χαρακτηριστικά Μη Σχεσιακών Βάσεων Δεδομένων

Τα βασικά χαρακτηριστικά των NoSQL database systems είναι:

1) Η μη χρήση της SQL ως γλώσσα ερωτημάτων (query language)

2) Δεν μπορεί να εγγυηθεί ότι οι διεργασίες στην βάση δεδομένων θα γίνονται αξιόπιστα

3) Έχουν μια ιδιαίτερη αρχιτεκτονική και φιλοσοφία λειτουργίας.

Αναλυτικότερα, τα NoSQL συστήματα αναπτύχθηκαν και εξελίχθηκαν παράλληλα με τις μεγαλύτερες εταιρίες πληροφορικής στον κόσμο, όπως η Google ή η Amazon, που είχαν ανάγκη να διαχειρίζονται έναν τεράστιο όγκο δεδομένων, κάτι το οποίο δεν εξυπηρετούσαν τα παραδοσιακά μέχρι τότε συστήματα RDBMS.

Τα NoSQL συστήματα έχουν φτιαχτεί έτσι ώστε να μπορούν να διαχειριστούν μεγάλες ποσότητες δεδομένων χωρίς κατ' ανάγκη να διατηρούν μία συγκεκριμένη δομή (schema). Επίσης, οι μέθοδοι υλοποίησης και εφαρμογής τους αξιοποιούν μια αρχιτεκτονική που επιτρέπει (και ίσως διευκολύνει) την κατανεμημένη λειτουργία του συστήματος. Έτσι με αυτόν τον τρόπο οι επιδόσεις του συστήματος μπορούν να αυξηθούν σημαντικά, από την στιγμή που μπορούν να προστεθούν θεωρητικά άπειροι servers όπου κατανεμημένα θα επεξεργάζονται τα δεδομένα του συστήματος. Ακόμη, η ενδεχόμενη αδυναμία λειτουργίας ενός server του συστήματος μπορεί να αντιμετωπισθεί εύκολα. Αυτός ο τύπος βάσεων δεδομένων αναπτύσσεται διαρκώς οριζόντια, καθώς αυξάνονται οι ανάγκες για storage δεδομένων, που συνεχώς πληθαίνουν (data growth), ενώ είναι πιο σημαντικές οι επιδόσεις προσπέλασης των δεδομένων από την συνοχή που παρουσιάζουν. [3]

1.4 Πλεονεκτήματα Μη Σχεσιακών Βάσεων Δεδομένων

- Είναι ανοιχτού κώδικα
- Προσφέρουν οριζόντια επεκτασιμότητα
- Υποστηρίζουν Map/Reduce
- Είναι απλές στη χρήση
- Παρέχουν μεγάλες ταχύτητες στην εισαγωγή δεδομένων και στις απλές λειτουργίες επεξεργασίας
- Η δομή δεδομένων (schema) μπορεί να αλλάξει χωρίς να επηρεαστεί το υπόλοιπο σύστημα
- Έχουν την ικανότητα να αποθηκεύουν περίπλοκα είδη δεδομένων

1.5 Μειονεκτήματα Μη Σχεσιακών Βάσεων Δεδομένων

- Θέματα διαχείρισης
- Δεν προσφέρουν δυνατότητα indexing

- Δεν παρέχουν λειτουργίες που βασίζονται στις ιδιότητες ACID
- Περίπλοκα μοντέλα συνέπειας/συνοχής
- Έλλειψη τυποποίησης (σε επίπεδο API και query language) [4]

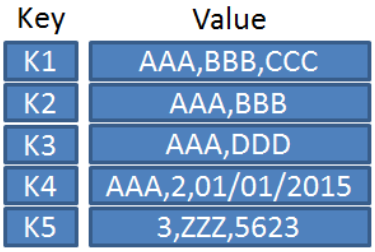
1.6 Κατηγορίες NoSQL Συστημάτων

Τα NoSQL συστήματα μπορούν να κατηγοριοποιηθούν ως εξής με βάση την αρχιτεκτονική τους και το μοντέλο δεδομένων που ακολουθούν: [4]

- Key-value stores
- Document stores
- Column stores

1.6.1 Key Value Stores

Τα δεδομένα αποθηκεύονται σαν ομάδες ζευγαριών κλειδιού-τιμής, τα οποία απαρτίζονται από δύο στοιχεία τα οποία συνδέονται μεταξύ τους. Η σύνδεση μεταξύ των στοιχείων αυτών, είναι ένα «κλειδί» το οποίο δρα ως αναγνωριστικό ενός στοιχείου δεδομένων και της τιμής του.



Εικόνα 2. Διαφορετικού τύπου δεδομένα τα οποία συσχετίζονται με το αντίστοιχο κλειδί. [9]

Στο συγκεκριμένο μοντέλο, τα δεδομένα αποθηκεύονται με τη μορφή hash πινάκων με ένα μοναδικό κλειδί και ένα δείκτη σε συγκεκριμένο αντικείμενο. Όπως και στους παραδοσιακούς πίνακες hash,η ανάκτηση και η αποθήκευση των δεδομένων γίνεται με τη βοήθεια των κλειδιών. Τα δεδομένα αυτά μπορεί να είναι οποιουδήποτε τύπου δυαδικά αντικείμενα όπως κείμενο, βίντεο, έγγραφα JSON κλπ.

Η εφαρμογή έχει πλήρη έλεγχο για το τι αποθηκεύεται ως τιμή, γεγονός το οποίο καθιστά τα key-value stores ως το πιο ευέλικτο NoSQL μοντέλο. Οι key-value βάσεις δεδομένων δίνουν εμφαση στην απλότητα και είναι χρήσιμες όταν μία εφαρμογή απαιτείται να υποστηρίζει λειτουργίες ανάγνωσης και γραφής δεδομένων με υψήλη επεξεργαστική ισχύ. A Survey Paper on NoSQL Databases: Key-Value Data Stores and Document Stores

Οι πιο γνωστές key-value βάσεις δεδομένων είναι οι εξής:

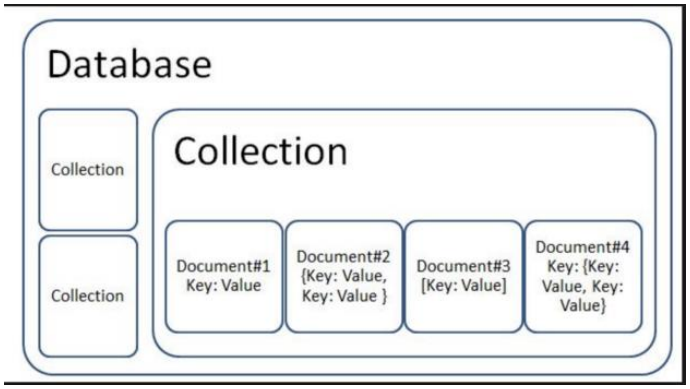
- Aerospike
- Apache Cassandra
- Amazon Dynamo DB
- Berkeley DB
- Couchbase
- Memcached
- Riak
- Redis

1.6.2 Document Stores

Τα document stores [6] θεωρούνται ως το επόμενο βήμα των key-value stores. Η διαφορά τους είναι ότι τα document stores περιλαμβάνουν πιο πολύπλοκα και δομημένα δεδομένα και επιτρέπουν την ενθυλάκωση των ζευγαριών κλειδιών-τιμής σε «έγγραφα». Το συγκεκριμένο μοντέλο, είναι πρακτικά ένα key value store, καθώς τα document ids αποτελούν το κλειδί (key) και το ίδιο το έγγραφο αποτελεί την τιμή (value). Σε αντίθεση με τις παραδοσιακές σχεσιακές βάσεις δεδομένων, τα δεδομένα σε ένα document store, δεν είναι δομημένα σε μορφή πινάκων με γραμμές και στήλες.

Αντί για στήλες με ονόματα και τύπους δεδομένων που χρησιμοποιούνται στις σχεσιακές βάσεις δεδομένων, ένα έγγραφο περιλαμβάνει μία περιγραφή του τύπου δεδομένων και την τιμή της περιγραφής αυτής. Κάθε έγγραφο μπορεί αν έχει την ίδια ή διαφορετική δομή από τα υπόλοιπα. Για την προσθήκη επιπλέον τύπων δεδομένων σε ένα document store, δεν απαιτείται η τροποποίηση ολόκληρου του συστήματος, αλλά η δημιουργία ένα αντικειμένου (object) στη βάση δεδομένων.

Τα έγγραφα ομαδοποιούνται σε συλλογές (collections), τα οποία ουσιαστικά είναι ότι και οι πίνακες στις σχεδιακές βάσεις δεδομένων. Τα document stores παρέχουν έναν μηχανισμό ερωτημάτων (query mechanism) για την αναζήτηση σε συλλογές με έγγραφα συγκεκριμένων ιδιοτήτων. [7]



Εικόνα 3. Document Stores

Τα κυριότερα προτερήματα των document stores είναι η ευελιξία στη δημιουργία των μοντέλων δεδομένων, η υψηλή ταχύτητα εγγραφής και η υψηλή αποδοτικότητα στα ερωτήματα (queries).

Τα document stores χρησιμοποιούνται κυρίως όταν απαιτείται ένα ευέλικτο schema. Η βάσεις δεδομένων εγγράφων είναι schema-less, που σημαίνει ότι τα έγγραφα μπορεί να αποτελούνται από διαφορετικού είδους schemas και τιμές δεδομένων, σε αντίθεση με το σχεσιακό μοντέλο όπου κάθε γραμμή σε έναν πίνακα θα πρέπει να διαθέτει τις ίδιες στήλες.

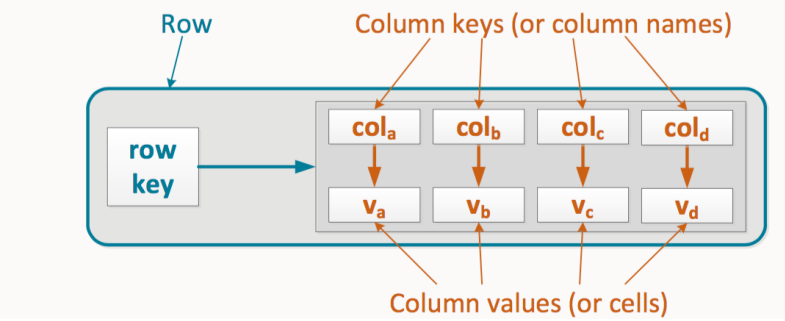
Οι πιο γνωστές μη σχεσιακές βάσεις δεδομένων που ακολουθούν το μοντέλο των document stores είναι η Apache CouchDB και η MongoDB.

1.6.3 Column Stores

Στα column stores [6] τα δεδομένα αποθηκεύονται σε κελιά (cells), τα οποία ομαδοποιούνται σε στήλες (columns) και όχι σε σειρές (rows) όπως στις σχεσιακές βάσεις δεδομένων. Οι στήλες αυτές με την σειρά τους ομαδοποιούνται σε οικογένειες στηλών (column families), οι οποίες μπορούν να περιλαμβάνουν θεωρητικά άπειρο αριθμό από στήλες. Με τον τρόπο αυτό επιτυγχάνονται μεγάλες

ταχύτητες σε λειτουργίες search/access, λόγω του ότι όλα τα κελιά που αναφέρονται σε μια στήλη είναι μια συνεχόμενη εγγραφή στο δίσκο.

Οι πιο γνωστές εφαρμογές του μοντέλου column store είναι το project BigTable της Google και το Apache Cassandra.



Εικόνα 4. Column Stores

1.6.4 Βάσεις Δεδομένων Γράφων

Το μοντέλο αυτό, χρησιμοποιεί δομές γράφων και είναι βασισμένη σε κόμβους (nodes) [6], τις σχέσεις μεταξύ αυτών των κόμβων και τις ιδιότητές τους. Αντί για πίνακες με στήλες και σειρές, εδώ υπάρχει ένα ευέλικτο γραφικό μοντέλο (graphmodel) που μπορεί να χρησιμοποιηθεί και να αναπτυχθεί παράλληλα σε πολλά μηχανήματα (servers – κόμβους).

Στον πίνακα που ακολουθεί, κατατάσσονται οι κατηγορίες των NoSQL συστημάτων ανάλογα με τα χαρακτηριστικά τους και τις βασικές τους ιδιότητες.

Μοντέλο	**Αποδοτικότητα**	**Επεκτασιμότητα**	**Ελαστικότητα**	**Πολυπλοκότητα**	**Λειτουργικότητα**
Key-Value store	Υψηλή	Υψηλή	Υψηλή	Καμία	Μεταβλητή (Καμία)
Column store	Υψηλή	Υψηλή	Μέτρια	Χαμηλή	Ελάχιστη
Document store	Υψηλή	Μεταβλητή (Υψηλή)	Υψηλή	Χαμηλή	Μεταβλητή (Χαμηλή)
Graph Database	Μεταβλητή	Μεταβλητή	Υψηλή	Υψηλή	Θεωρία Γράφων

Πίνακας 1. Βασικές ιδιότητες των διαφόρων μοντέλων [5]

1.7 Επίλογος

Στην παρούσα πτυχιακή εργασία, επιλέχθηκε η χρήση του document store μοντέλου και η MongoDB για την σχεδίαση ενός CMS. Στα επόμενα κεφάλαια θα εμβαθύνουμε περισσότερο στις τεχνικές λεπτομέρειες της MongoDB.

Κεφάλαιο 2ο: MongoDB

2.1 Εισαγωγή

Η MongoDB είναι μία ανοιχτού κώδικα, μη σχεσιακή βάση δεδομένων, η οποία ανήκει στην κατηγορία των NoSQL βάσεων δεδομένων. Αυτό σημαίνει ότι η αποθήκευση των δεδομένων δεν γίνεται σε πίνακες, όπως στις παραδοσιακές σχεσιακές βάσεις δεδομένων, αλλά στην περίπτωση της MongoDB τα δεδομένα αποθηκεύονται σε σχήματα (schemas) με μορφή παρόμοια με αυτή του προτύπου JSON (JavaScript Object Notation).

Πιο συγκεκριμένα, η MongoDB αποθηκεύει τα δεδομένα σε έγγραφα χρησιμοποιώντας μία δυαδική αναπαράσταση, που ονομάζεται BSON (Binary JSON). Έτσι, δεν χρειάζεται να χρησιμοποιούνται πίνακες, όπως την περίπτωση των σχεσιακών βάσεων, όπου ο κάθε πίνακας έχει συγκεκριμένο αριθμό πεδίων και οι εγγραφές μπορεί να περιέχουν πολλά κενά εξαιτίας του σταθερού αριθμού πεδίων του πίνακα. Με λίγα λόγια, το schema της βάσης δεν είναι απαραίτητα σταθερό.

Η MongoDB είναι μία βάση δεδομένων που αποτελείται από μία συλλογή εγγράφων στα οποία αποθηκεύονται οι απαιτούμενες πληροφορίες. Στην MongoDB σχετικά μεταξύ τους δεδομένα αποθηκεύονται μαζί για την γρήγορη απάντηση των ερωτημάτων μέσω της γλώσσας ερωτημάτων της MongoDB. Έτσι, διαφορετικά document μεταξύ τους μπορεί να έχουν διαφορετική μορφή.

Επιπρόσθετα, δεν είναι αναγκαία η δήλωση της μορφής των εγγράφων στο σύστημα, αφού κάθε έγγραφο είναι περιγραφικό ως προς τη δομή του. Τέλος, θα πρέπει να προσθέσουμε ότι είναι δυνατόν να προστεθούν πεδία σε κάποιο έγγραφο ανά πάσα στιγμή, χωρίς αυτό να επηρεάσει τα υπόλοιπα έγγραφα, ούτε και τις υπάρχουσες εγγραφές.

Η μορφή των δεδομένων που αποθηκεύονται σε μία βάση δεδομένων MongoDB ποικίλει. Μπορεί να είναι είτε έγγραφα είτε κάποια δομή δεδομένων, ενώ είναι δυνατή η αλλαγή τους με την πάροδο του χρόνου. Τα δεδομένα που αποθηκεύονται συνήθως αντιστοιχούν σε αντικείμενα στον κώδικα της εφαρμογής, κάνοντας τα ευέλικτα στην επεξεργασία.

Τα ερωτήματα προς μία βάση δεδομένων MongoDB, για τη συλλογή δεδομένων είναι δυναμικά και δεν απαιτούν συγκεκριμένους δείκτες. Για τα ερωτήματα χρησιμοποιείται η MongoDB's document-based query language. Στα ερωτήματα αυτά δεν περιέχονται join, καθώς δεν θα είναι εφικτό το scaling. Η mongoDB, όσον αφορά τα δεδομένα πουαποθηκεύονται, είναι κατάλληλη για χρήση σε εφαρμογές όπου υπάρχει μεγάλος όγκος δεδομένων, τα οποία αλλάζουν πολύ γρήγορα και απαιτείται ταχύτητα.

Όπως αναφέραμε και στην ενότητα όπου εξετάστηκαν τα βασικά χαρακτηριστικά των document stores, η MongoDB αποθηκεύει τα δεδομένα σε έγγραφα. Κάθε έγγραφο αποτελεί ένας ζεύγος κλειδιών-τιμών (key-value pairs). Κάθε έγγραφο μπορεί να έχει μέγεθος έως και 16MB. Επιπλέον, κάθε έγγραφο έχει ένα μοναδικό _id, με βάση το οποίο δεικτοδοτείται (indexed) αυτόματα. Η τιμή του πεδίου _id παράγεται αυτόματα, αν δεν έχει οριστεί εξ΄ αρχής και χρησιμοποιείται ως το μοναδικό αναγνωριστικό του εγγράφου.

Τέλος, θα πρέπει να αναφέρουμε ότι πολλαπλά έγγραφα οργανώνονται σε συλλογές (collections), οι οποίες είναι αντίστοιχες των πινάκων στις σχεσιακές βάσεις δεδομένων. Οι συλλογές μπορούν να περιέχουν έγγραφα με διαφορετική δομή ως προς το σχήμα (schema), ωστόσο είναι καλή πρακτική να αποθηκεύονται έγγραφα με την ίδια δομή.

Εικόνα 5. Παράδειγμα ενός document

Η βάση δεδομένων της MongoDB είναι διαθέσιμη σε 2 εκδόσεις, την Community και την Enterprise. Η MongoDB Community είναι μία ανοιχτού κώδικα έκδοση η οποία είναι διαθέσιμη δωρεάν. Η MongoDB Enterprise διατίθεται ως μέρος της συνδρομητικής MongoDB Enterprise Advanced και παρέχει υποστήριξη για deployment. Παρέχει επίσης δυνατότητες όπως το LDAP και το Kerberos, on-disk κρυπτογράφηση και συστήματα ελέγχου (auditing). [1]

2.2 Έγγραφα

Όπως αναφέραμε και παραπάνω, η MongoDB, αποθηκεύει εγγραφές δεδομένων με τη μορφή BSON εγγράφων (documents), τα οποία είναι η δυαδική απεικόνιση των JSON documents με τη διαφορά του ότι περιέχουν περισσότερους τύπους δεδομένων σε σχέση με τα JSON.

2.2.1 Δομή των Εγγράφων

Τα documents στην MongoDB αποτελούνται από ζευγάρια πεδίου και τιμής και έχουν την ακόλουθη δομή [10]:

```
{
   field1: value1,
   field2: value2,
   field3: value3,
   ...
   fieldN: valueN
}
```

Οι τιμές των πεδίων μπορεί να είναι οποιουδήποτε τύπου που συμπεριλαμβάνεται στους τύπους δεδομένων των BSON. Μπορεί επίσης οι τιμές των πεδίων να περιέχουν άλλα επιμέρους documents, πίνακες και πίνακες από documents.

Παράδειγμα:

```
var mydoc = {
               _id: ObjectId("5099803df3f4948bd2f98391"),
               name: { first: "Alan", last: "Turing" },
               birth: new Date('Jun 23, 1912'),
               death: new Date('Jun 07, 1954'),
               contribs:   [   "Turing   machine",   "Turing   test",
"Turingery" ],
               views : NumberLong(1250000)
            }
```

Τα παραπάνω πεδία περιλαμβάνουν τους εξής τύπους δεδομένων:

- Το `_id` συμπεριλαμβάνει ένα ObjectId

- Το `name` περιέχει ένα ενσωματωμένο έγγραφο (embedded document) το οποίο διαθέτει τα πεδία first και last.
- Τα `birth` και `death` είναι τύπου Date.
- Το `contribs` περιλαμβάνει ένα πίνακα από strings
- Το views είναι τύπου `NumberLong`

2.2.2 Ονόματα Πεδίων

Τα ονόματα των πεδίων είναι strings. Παρόλα αυτά, τα documents έχουν τους παρακάτω περιορισμούς όσον αφορά τα ονόματα των πεδίων [10]:

- Το πεδίο _id είναι δεσμευμένο και χρησιμοποιείται ως κύριο κλειδί. Η τιμή του είναι μοναδική μέσα σε ένα collection, δεν μπορεί να δεχθεί περαιτέρω αλλαγές μετά τη δημιουργία του και μπορεί να είναι οποιουδήποτε τύπου εκτός από array.
- Τα ονόματα των πεδίων δεν μπορούν να περιέχουν τον χαρακτήρα null
- Δεν μπορούν να ξεκινάνε με τον χαρακτήρα "$"
- Δεν μπορούν να ξεκινάνε με τελεία (.)

2.2.3 Πλεονεκτήματα των εγγράφων

Τα πλεονεκτήματα ενός εγγράφου είναι τα παρακάτω:

- Τα έγγραφα μπορούν να αναπαρασταθούν εύκολα σε κάθε γλώσσα προγραμματισμού με τους τύπους δεδομένων που διαθέτει η κάθε μια.
- Τα ενσωματωμένα έγγραφα και οι πίνακες μειώνουν την ανάγκη για περίπλοκες συνενώσεις (expensive joins).
- Τα έγγραφα δεν έχουν στατική μορφή και μπορούν να αλλάξουν οποιαδήποτε στιγμή δυναμικά, υποστηρίζοντας έτσι ευέλικτα τον πολυμορφισμό και δίνοντας μια ευελιξία στο προγραμματιστή για τη δομή της βάσης δεδομένων.

2.3 Χαρακτηριστικά της MongoDB

Η MongoDB προσφέρει τα ακόλουθα χαρακτηριστικά:

- **Ad hoc ερωτήματα:** Η MongoDB υποστηρίζει πεδία, ερωτήματα και regular expressions για αναζήτηση δεδομένων. Τα ερωτήματα μπορούν να επιστρέφουν συγκεκριμένα πεδία των εγγράφων και μπορεί επίσης, να περιλαμβάνουν συναρτήσεις που ορίζονται από συναρτήσεις της JavaScript. Τα ερωτήματα μπορεί επίσης να ρυθμιστούν να επιστρέφουν ένα τυχαίο δείγμα αποτελεσμάτων ενός δεδομένου μεγέθους.
- **Indexing:** Τα πεδία σε ένα έγγραφο MongoDB μπορούν να πάρουν την μορφή ευρετηρίου χρησιμοποιώντας πρωτογενείς και δευτερεύοντες δείκτες. Η κατηγοριοποίηση των δεδομένων σε πραγματικό χρόνο, δίνει τη δυνατότητα και τα εργαλεία για πρόσβαση και ανάλυση των δεδομένων.
- **Replication:** Το MongoDB παρέχει υψηλή διαθεσιμότητα μέσω αντιγράφων (replica sets). Ένα replica set αποτελείται από δύο ή περισσότερα αντίγραφα δεδομένων. Κάθε replica set μπορεί να έχει το ρόλο πρωτεύοντος ή δευτερεύοντος αντιγράφου κάθε στιγμή. Η εγγραφή και η ανάγνωση δεδομένων γίνονται στο πρωτεύων αντίγραφο. Τα δευτερεύοντα αντίγραφα διατηρούν ένα αντίγραφο των πρωτεύων δεδομένων μια τεχνική αντιγραφής. Όταν αποτύχει ένα πρωτεύον αντίγραφο, διεξάγει αυτόματα μια εκλογική διαδικασία στο σύνολο των replica sets για να προσδιοριστεί ποιο δευτερεύον replica set πρέπει να γίνει το κύριο.Τα δευτερεύοντα τμήματα μπορούν προαιρετικά να προσφέρουν λειτουργίες ανάγνωσης, αλλά αυτά τα δεδομένα είναι συνεπή.

- **Load balancing:** Η MongoDB προσφέρει οριζόντια κλιμάκωση με τη χρήση sharding. Ο χρήστης επιλέγει ένα sharded κλειδί, το οποίο καθορίζει τον τρόπο με τον οποίο θα διανεμηθούν τα δεδομένα μιας συλλογή. Τα δεδομένα χωρίζονται σε εύρη τιμών (με βάση το sharded κλειδί) και κατανέμονται μεταξύ πολλαπλών shards. Ένα shard είναι master με έναν ή περισσότερους slaves. Εναλλακτικά, το shard κλειδί μπορεί να επιτρέψει μια ομοιόμορφη κατανομή δεδομένων. Η MongoDB μπορεί να τρέχει παράλληλα σε πολλαπλούς servers, εξισορροπώντας το φορτίο ή αντιγράφοντας τα δεδομένα για να διατηρήσει το σύστημα σε λειτουργία σε περίπτωση αποτυχίας κάποιου υλικού (hardware).
- **Αποθήκευση αρχείων (File Storage):** Η MongoDB μπορεί να χρησιμοποιηθεί σαν ένα σύστημα αρχείων με λειτουργίες εξισορρόπησης φορτίου και αντιγραφής δεδομένων σε πολλαπλά μηχανήματα για την αποθήκευση αρχείων. Η λειτουργία αυτή ονομάζεται grid file system. Η MongoDB διαθέτει λειτουργίες επεξεργασίας των αρχείων και του περιεχομένου τους που μπορούν να χρησιμοποιηθούν από τους προγραμματιστές.
- **Aggregation:** Το MapReduce μπορεί να χρησιμοποιηθεί για την επεξεργασία δεδομένων σε ομάδες και λειτουργίες συγκέντρωσης (aggregation). Η λειτουργία aggregation δίνει την δυνατότητα στους χρήστες να πάρουν αποτελέσματα παρόμοια με αυτά που προκύπτουν από το GROUP BY της SQL. Επιπλέον, η λειτουργία aggregation δίνει την δυνατότητα σχηματισμού pipeline αντίστοιχο του Unix και προσφέρει τη λειτουργία $lookup για τη συνένωση εγγράφων από πολλαπλά έγγραφα.
- **Χρήση JavaScript στην πλευρά του server:** Η JavaScript μπορεί να χρησιμοποιηθεί σε ερωτήματα, aggregation συναρτήσεις που αποστέλλονται απευθείας στη βάση δεδομένων προς εκτέλεση.
- **Capped collections:** Η MongoDB υποστηρίζει συλλογές (collections) σταθερού μεγέθους που ονομάζονται capped collections. Αυτός ο τύπος συλλογής επιτρέπει την εισαγωγή δεδομένων με τη σειρά και όταν φτάσει σε ένα καθορισμένο μέγεθος, συμπεριφέρεται σαν circular queue.
- **Υψηλή Αποδοτικότητα:** Η MongoDB παρέχει υψηλή αποδοτικότητα και συγκεκριμένα υποστηρίζει ενσωματωμένα (embedded) μοντέλα δεδομένων τα οποία βοηθούν στην μείωση της I/O (Input/Output) δραστηριότητας σε ένα σύστημα βάσης δεδομένων.
- **Οριζόντια Επεκτασιμότητα:** Με τη χρήση της MongoDB, παρέχεται αυτομάτως και οριζόντια επεκτασιμότητα, η οποία αποτελεί ένα μεγάλο μέρος της κύριας λειτουργικότητας της MongoDB. Η μέθοδος Sharding, βοηθά στη διανομή των δεδομένων ανάμεσα σε ένα cluster διαφόρων συστημάτων. Ξεκινώντας από την έκδοση 3.4, η MongoDB υποστηρίζει επίσης τη δημιουργία διαφόρων ζωνών (zones) από δεδομένα βασισμένα στο sharding key. Σε ένα εξισορροπημένο cluster, η ανάγνωση και η εγγραφή των δεδομένων που καλύπτονται από ένα zone, καθοδηγείται μόνο μέσα στα shards που περιέχονται μέσα στη ζώνη αυτή.
- **Πολλαπλά συστήματα αποθήκευσης:** Η MongoDB υποστηρίζει πολλά συστήματα αποθήκευσης όπως το WiredTiger Storage Engine και το In-Memory Storage Engine. Επιπρόσθετα, παρέχεται η σύνδεση με συγκεκριμένο API το οποίο επιτρέπει την ανάπτυξη ξεχωριστών και νέων συστημάτων αποθήκευσης από third party προγραμματιστές.

2.4 Εγκατάσταση της MongoDB σε Windows:

Για να εγκαταστήσουμε δωρεάν την MongoDB σε υπολογιστή ο οποίος διαθέτει λειτουργικό σύστημα Windows, κατεβάζουμε την τελευταία της έκδοση από το επίσημο site, αφού επιλέξουμε την έκδοση MongoDB Community Server.

Έπειτα παρατηρούμε ότι όταν εκτελέσουμε την εγκατάσταση του αρχείου που κατεβάσαμε, αυτό θα εγκατασταθεί από προεπιλογή στον φάκελο C:\Program Files\.

Η MongoDB χρειάζεται έναν φάκελο στον οποίο θα αποθηκεύει όλα της τα αρχεία. Ο προεπιλεγμένος φάκελος για τα δεδομένα της MongoDB είναι το directory c:\data\db.

Για να δημιουργήσουμε τον φάκελο, γράφουμε τις εξής εντολές στη Γραμμή Εντολών των Windows:

```
C:\>md data
C:\:md data\db
```

Στη συνέχεια, θα πρέπει να προσδιορίσουμε το dbpath στο mongod.exe με τις ακόλουθες εντολές:

```
C:\Users\XYZ>d:cd C:\Program Files\MongoDB\Server\4.2\bin
C:\Program Files\MongoDB\Server\4.2\bin>mongod.exe --dbpath
"C:\data"
```

Μετά την εκτέλεση των παραπάνω εντολών, θα εμφανιστεί το μήνυμα “Waiting for connections” στο console output, το οποίο δείχνει ότι η διεργασία mongod.exe έχει ξεκινήσει επιτυχώς και εκείνη τη στιγμή εκτελείται.

Για να ξεκινήσουμε την MongoDB, θα χρειαστεί να ανοίξουμε ένα δεύτερο παράθυρο γραμμής εντολών (χωρίς να κλείσει το πρώτο) στο οποίο γράφουμε την παρακάτω εντολή:

```
C:\Program Files\MongoDB\Server\4.2\bin>mongo.exe
```

η οποία θα εμφανίσει το ακόλουθο output:

```
MongoDB shell version v4.2.1
connecting to:
mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName
=mongodb
Implicit session: session { "id" : UUID("4260beda-f662-4cbe-9bc7-
5c1f2242663c") }
MongoDB server version: 4.2.1
```

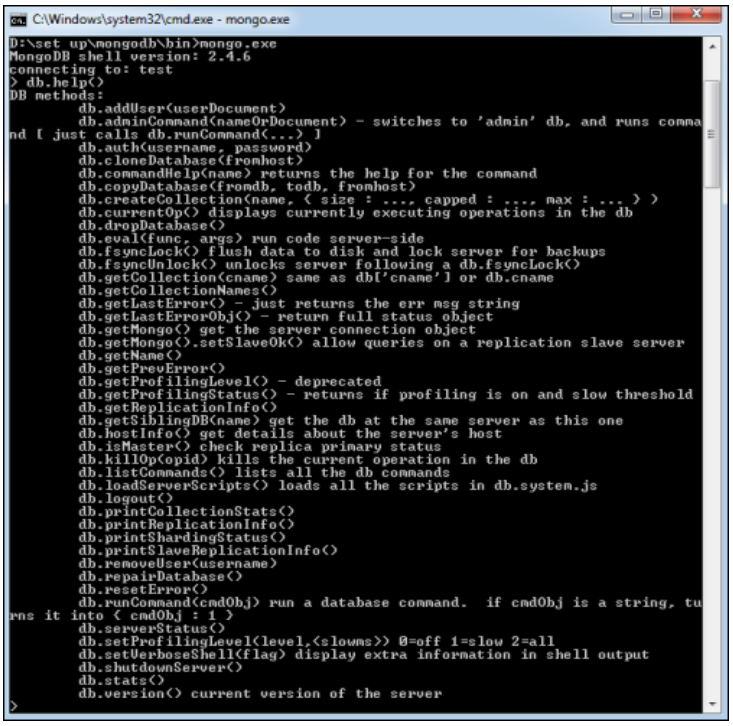
Βλέποντας το παραπάνω, καταλαβαίνουμε ότι η MongoDB έχει εγκατασταθεί και η διεργασία της εκτελείται.

Παρόλα αυτά, τα παραπάνω βήματα αναφέρονται μόνο στην πρώτη φορά που εγκαθιστούμε και τρέχουμε την MongoDB. Τις επόμενες φορές, αρκούν μόνο οι παρακάτω δύο εντολές:

```
C:\Program Files\MongoDB\Server\4.2\bin>mongod.exe --dbpath
"C:\data"
C:\Program Files\MongoDB\Server\4.2\bin>mongo.exe
[19]
```

2.4.1 MongoDB Help

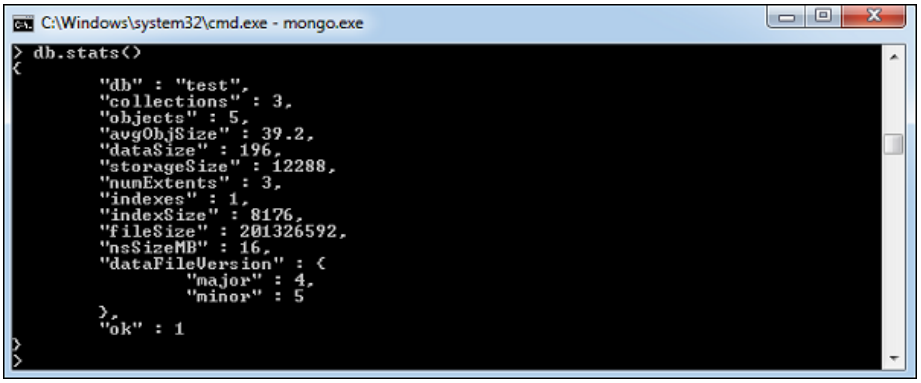
Για να εμφανίσουμε τη λίστα των εντολών που υποστηρίζει η MongoDB, πληκτρολογούμε την εντολή db.help(), της οποίας το αποτέλεσμα θα μοιάζει με το παρακάτω:



Εικόνα 6. Mongo Help [10]

2.4.2 MongoDB Statistics

Για να δούμε τα στατιστικά για τον MongoDB server, πληκτρολογούμε την εντολή db.stats(). Παρατηρούμε ότι εμφανίζεται το όνομα της βάσης δεδομένων, ο αριθμός των collections και των documents και κάποιες περαιτέρω πληροφορίες σχετικά με τα μεγέθη των αρχείων: [10]



Εικόνα 7. MongoDB Statistics

2.4.3 Mongo shell (Κέλυφος)

Εγκαθιστώντας τοπικά τη MongoDB εγκαθίσταται και ένα κέλυφος (shell), μέσα από το οποίο μπορούμε να εκτελέσουμε διάφορα ερωτήματα. [10]
Για να ανοίξουμε ένα mongo κέλυφος αρκεί σε ένα παράθυρο γραμμής εντολών να πληκτρολογήσουμε:
`>>>mongo`

Εφόσον συνδεθούμε στο κέλυφος μπορούμε να δούμε όλες τις διαθέσιμες βάσεις δεδομένων που υπάρχουν πληκτρολογώντας:

```
>>>show databases
```

Για να δημιουργήσουμε μια νέα αρκεί να εισάγουμε το παρακάτω:

```
>>>use db_name
```

Η εντολή `use <db>` αυτό που κάνει είναι να επιλέξει τη συγκεκριμένη βάση ως τρέχον βάση και σε περίπτωση που αυτή δεν υπάρχει τη δημιουργεί.

2.5 Συλλογές (Collections)

Στη MongoDB ένα σύνολο εγγράφων αποτελούν μια Συλλογή (Collection) η οποία σε αντιστοίχιση με τις σχεσιακές βάσεις δεδομένων μπορούμε να πούμε ότι είναι ένας πίνακας (table).

Για να δημιουργήσουμε μια νέα Συλλογή πληκτρολογούμε στο mongo κέλυφος το παρακάτω:

```
>>>db.myNewCollection.insertOne( { name: "John" } )
```

Εάν πριν την παραπάνω εντολή δεν έχουμε επιλέξει μία ήδη υπάρχουσα βάση, τότε χρησιμοποιείται η test.

Το όνομα της Συλλογής που δημιουργήθηκε με τη παραπάνω εντολή είναι το myNewCollection. Παρατηρούμε ότι πρέπει να αποθηκεύσουμε ένα Έγγραφο στη Συλλογή για να δημιουργηθεί. Στην παραπάνω περίπτωση αυτό επιτυγχάνεται με την μέθοδο insertOne(). Επίσης διατίθεται και η insertMany() η οποία μας επιτρέπει να εισάγουμε παραπάνω από ένα Έγγραφα σε μια εντολή. Οι εντολές δημιουργίας και εισαγωγής θα περιγραφούν αναλυτικά σε επόμενη ενότητα.

Μπορούμε να δούμε όλες τις διαθέσιμες Συλλογές που υπάρχουν σε μια MongoDB βάση δίνοντας τη παρακάτω εντολή στο shell:

```
>>>show collections
```

Μπορούμε να δούμε όλα τα Έγγραφα που περιέχει μια Συλλογή δίνοντας:

```
>>>db.myNewCollection.find().pretty()
```

Η μέθοδος find() αυτό που κάνει είναι να εκτελεί ένα ερώτημα και να βρίσκει Έγγραφα με συγκεκριμένα κριτήρια, τα οποία περνιούνται σαν παράμετροι. Όταν δεν υπάρχουν παράμετροι επιστρέφει όλα τα Έγγραφα από μια συλλογή.

Η μέθοδος pretty() παρουσιάζει το αποτέλεσμα του ερωτήματος πιο ευανάγνωστο στην οθόνη. [10]

2.6 Βασικές Λειτουργίες (CRUD Operations)

Στην ενότητα αυτή θα περιγράψουμε τα CRUD Operations (Create, Read, Update, Delete λειτουργίες) της MongoDB [11]. Αναφέρονται ενδεικτικά οι εντολές που μπορούν να γραφτούν σε Mongo Shell. Εναλλακτικά, οι αντίστοιχες εντολές και λειτουργίες μπορούν να γραφτούν και σε Python, Java (σύγχρονη), Java (ασύγχρονη) Nodejs, php, Motor, C#, Perl, Ruby, Scala κλπ. Αναλυτικές οδηγίες για

τη μετάφραση των εντολών στην κάθε γλώσσα προγραμματισμού, υπάρχουν στο επίσημο documentation της MongoDB. [10]

2.6.1 Δημιουργία (Insert)

Η εντολή db.collection.InsertOne() εισάγει ένα απλό document μέσα σε ένα collection.

Για παράδειγμα, στο ακόλουθο κομμάτι κώδικα, εισάγεται ένα νέο document μέσα σε ένα collection που ονομάζεται products.

```
try {
   db.products.insertOne( { item: "card", qty: 15 } );
} catch (e) {
   print (e);
};
```

Έπειτα από τη διαδικασία αυτή, στο shell τυπώνεται το ακόλουθο output:

```
{
   "acknowledged" : true,
   "insertedId" : ObjectId("56fc40f9d735c28df206d078")
}
```

Στην MongoDB, κάθε document που είναι αποθηκευμένο σε ένα collection, χρειάζεται ένα μοναδικό πεδίο _id, το οποίο πρακτικά λειτουργεί σαν μοναδικό κλειδί (primary key). Εάν το πεδίο _id δεν οριστεί κατά τη διαδικασία του insert, τότε ο MongoDB driver δημιουργεί και παράγει μία ObjectId τιμή για το πεδίο αυτό.

Τα ObjectIds είναι μικρά σε μήκος, συνήθως μοναδικά, εύκολα στη δημιουργία και ταξινομημένα. Οι τιμή ενός ObjectId έχουν μήκος 12 bytes και αποτελούνται από:

- 4 bytes τα οποία φανερώνουν τη στιγμή (σε δευτερόλεπτα) (timestamp) που δημιουργήθηκε το ObjectId
- 5 bytes με τυχαία τιμή
- 3 bytes για τον αύξοντα αριθμό (συνήθως αρχίζει από μία τυχαία τιμή και αυξάνεται κατά ένα κάθε φορά)

Παρόλα αυτά, ένας εναλλακτικός τρόπος ορισμού της τιμής του πεδίου _id είναι να οριστεί χειροκίνητα, κατά τη στιγμή δημιουργίας του document. Στο ακόλουθο παράδειγμα βλέπουμε ότι η μέθοδος insertOne() περιέχει και τον ορισμό του _id. Το _id, πρέπει να είναι μοναδικό μέσα σε ένα collection για την αποφυγή των duplicate key errors. Για το λόγο αυτό, στο παράδειγμα που ακολουθεί, η insertOne περικλείεται από μία try-catch.

```
try {
   db.products.insertOne( { _id: 10, item: "box", qty: 20 } );
} catch (e) {
   print (e);
}
```

Η διαδικασία αυτή, επιστρέφει το εξής αποτέλεσμα:

```
{ "acknowledged" : true, "insertedId" : 10 }
```

Εάν το πεδίο _id πάρει μία τιμή η οποία υπάρχει ήδη μέσα στο ίδιο collection, τότε επιστρέφεται το εξής exception:

```
WriteError({
   "index" : 0,
   "code" : 11000,
   "errmsg" : "E11000 duplicate key error collection:
inventory.products index: _id_ dup key: { : 10.0 }",
   "op" : {
      "_id" : 10,
      "item" : " box",
      "qty" : 20
   }
})
```

Στον ακόλουθο πίνακα παραθέτουμε συγκεντρωτικά τις πιο συνηθισμένες και βασικές insert μεθόδους:

`db.collection.insertOne()`	Εισάγει ένα document σε ένα collection
`db.collection.insertMany()`	Εισάγει πολλαπλά documents σε ένα collection
`db.collection.insert()`	Εισάγει είτε ένα είτε πολλαπλά documents σε ένα collection

Πίνακας 2. Insert μέθοδοι της MongoDB [11]

2.6.2 Αναζήτηση (Query)

Η αναζήτηση διαφόρων εγγράφων (documents) στην MongoDB [11], γίνεται με τη μέθοδο `db.collection.find()`.

Για την προβολή όλων των documents που εμπεριέχονται σε ένα collection, για παράδειγμα inventory, μπορεί να χρησιμοποιηθεί η εντολή `db.inventory.find( {} )` , η οποία αντιστοιχίζεται με το SELECT * FROM inventory που χρησιμοποιείται στην SQL.

Για τον ορισμό κριτηρίων ισότητας, χρησιμοποιείται η έκφραση <field>:<value>. Στο ακόλουθο παράδειγμα, επιλέγονται από το inventory collection, όλα τα documents των οποίων το πεδίο status, ισούται με "D":

```
db.inventory.find( { status: "D" } )
```

Η λειτουργία αυτή, είναι η αντίστοιχη του SQL statement SELECT * FROM inventory WHERE status = "D"

Εναλλακτικά, μπορούν να προστεθούν και query operators (τελεστές ερωτημάτων) στην αναζήτηση ενός document. Στο παράδειγμα που ακολουθεί, επιλέγονται όλα τα documents από το inventory collection, των οποίων το πεδίο status ισούται είτε με "A" ή "D":

```
db.inventory.find( { status: { $in: [ "A", "D" ] } } )
```

2.6.2.1 Τελεστές ερωτημάτων (Query Operators)

Η MongoDB περιέχει ένα σύνολο από τελεστές που βοηθάνε στην εκτέλεση ερωτημάτων. Κάποιοι από τους πιο γνωστούς τελεστές που χρησιμοποιούνται συχνότερα είναι:

- **$eq :** Αντιστοιχεί τις τιμές που είναι ίδιες με μια καθορισμένη τιμή.
- **$gt :** Αντιστοιχεί τις τιμές που είναι μεγαλύτερες από μια καθορισμένη τιμή.
- **$gte :** Αντιστοιχεί τις τιμές που είναι μεγαλύτερες-ίσες από μια καθορισμένη τιμή.
- **$lt :** Αντιστοιχεί τις τιμές που είναι μικρότερες από μια καθορισμένη τιμή.
- **$lte :** Αντιστοιχεί τις τιμές που είναι μικρότερες-ίσες από μια καθορισμένη τιμή.
- **$ne :** Αντιστοιχεί τις τιμές που δεν είναι ίδιες με μια καθορισμένη τιμή.
- **$in :** Αντιστοιχεί τις τιμές που βρίσκονται σε έναν πίνακα.
- **$nin :** Αντιστοιχεί τις τιμές που δεν βρίσκονται σε έναν πίνακα.
- **$or :** Συνενώνει ένα κριτήριο με ένα άλλο και επιστρέφει τα Έγγραφα που ταιριάζουν σε ένα από τα 2 κριτήρια.
- **$and :** Συνενώνει ένα κριτήριο με ένα άλλο και επιστρέφει τα Έγγραφα που ταιριάζουν και στις 2 συνθήκες.
- **$not :** Αντιστρέφει το ερώτημα και επιστρέφει Έγγραφα που δεν ταιριάζουν στα κριτήρια.
- **$exists :** Αντιστοιχεί Έγγραφα που έχουν τα καθορισμένα πεδία.
- **$all :** Αντιστοιχεί πίνακες που περιέχουν όλα τα στοιχεία του ερωτήματος.
- **$size :** Επιλέγει τα Έγγραφα που το μέγεθος του πεδίου του πίνακα τους έχει μια συγκεκριμένη τιμή.

2.6.3 Ενημέρωση (Update)

Για την ενημέρωση ενός document, η MongoDB [10] παρέχει update operators, όπως το $set, μέσω του οποίου μπορεί να ενημερωθεί η τιμή ενός πεδίου.

Για να χρησιμοποιήσουμε τους update operators, βάζουμε ένα update document μέσα σε μία update μέθοδο που μπορεί να έχει την παρακάτω μορφή:

```
{
  <update operator>: { <field1>: <value1>, ... },
  <update operator>: { <field2>: <value2>, ... },
  ...
}
```

Κάποιοι update operators, όπως για παράδειγμα το $set, δημιουργούν το αντίστοιχο πεδίο, εάν αυτό δεν υπάρχει ήδη.

2.6.3.1 Ενημέρωση ενός document

Στο ακόλουθο παράδειγμα, χρησιμοποιείται η μέθοδος db.collection.updateOne() για να ενημερώσει την τιμή του πρώτου document που θα βρεθεί, στο οποίο το πεδίο item έχει την τιμή “paper”:

```
db.inventory.updateOne(
   { item: "paper" },
   {
     $set: { "size.uom": "cm", status: "P" },
     $currentDate: { lastModified: true }
   }
)
```

Στο παράδειγμα αυτό, η λειτουργία update:

- χρησιμοποιεί τον operator $set για να θέσει την τιμή του πεδίου size.uom σε “cm” και την τιμή του πεδίου status σε “P”
- χρησιμοποιεί τον operator $currentDate για να θέσει την τιμή του πεδίου lastModified στην τρέχουσα ημερομηνία. Εάν το πεδίο lastModified δεν υπάρχει, τότε θα δημιουργηθεί από το $currentDate operator. [11]

2.6.3.2 Ενημέρωση πολλαπλών documents

Στο παράδειγμα που ακολουθεί, χρησιμοποιείται η μέθοδος db.collection.updateMany() στο inventory collection, για να ενημερώσει όλα τα documets, των οποίων το πεδίο qty έχει τιμή μικρότερη του 50:

```
db.inventory.updateMany(
   { "qty": { $lt: 50 } },
   {
     $set: { "size.uom": "in", status: "P" },
     $currentDate: { lastModified: true }
   }
)
```

Στο παράδειγμα αυτό, η λειτουργία update:

- χρησιμοποιεί τον operator $set για να θέσει την τιμή του πεδίου size.uom σε “in” και την τιμή του πεδίου status σε “P”

- χρησιμοποιεί τον operator $currentDate για να θέσει την τιμή του πεδίου lastModified στην τρέχουσα ημερομηνία. Εάν το πεδίο lastModified δεν υπάρχει, τότε θα δημιουργηθεί από το $currentDate operator.

Σε αυτό το σημείο, είναι σημαντικό να σημειώσουμε ότι οι τιμές των πεδίων _id, δεν μπορούν να αλλάξουν, αφού δημιουργηθούν. [11]

2.6.3.3 Αντικατάσταση ενός document

Για να αντικαταστήσουμε ολόκληρο το περιεχόμενο ενός document, εκτός από το πεδίο _id, ορίζουμε το νέο document σαν δεύτερο argument στη μέθοδο db.collection.replaceOne().

Όταν αντικαθιστούμε ένα document, το νέο document θα πρέπει να αποτελείται μόνο από ζευγάρια πεδίου και τιμής και δεν πρέπει να περιέχει εκφράσεις με update operators (πχ $set κλπ).

Το document που θα αντικαταστήσει το παλιό, μπορεί να περιέχει διαφορετικά πεδία.

Το παρακάτω παράδειγμα, αντικαθιστά το πρώτο document από το inventory collection, του οποίου το πεδίου item ισούται με “paper”:

```
db.inventory.replaceOne(
   { item: "paper" },
   { item: "paper", instock: [ { warehouse: "A", qty: 60 }, { warehouse: "B", qty: 40 } ] }
)
```

Στον παρακάτω πίνακα, παραθέτουμε τις βασικές μεθόδους ενημέρωσης/αντικατάστασης ενός εγγράφου σε ένα collection:

`db.collection.updateOne()`	Ενημερώνει το πολύ ένα document το οποίο πληροί τα φίλτρα που έχουν εφαρμοστεί, ακόμα και αν υπάρχουν πολλά documents που πληρούν τα κριτήρια αυτά.
`db.collection.updateMany()`	Ενημερώνει όλα τα documents που ταιριάζουν στα φίλτρα που έχουν εφαρμοστεί.
`db.collection.replaceOne()`	Αντικαθιστά το πολύ ένα document που πληροί τα αντίστοιχα κριτήρια, ακόμα και αν υπάρχουν πολλά documents που πληρούν τα κριτήρια αυτά
`db.collection.update()`	Ενημερώνει ή αντικαθιστά είτε ένα document, είτε πολλά, τα οποία ταιριάζουν στα φίλτρα που έχουν εφαρμοστεί

Πίνακας 3. Update μέθοδοι της MongoDB [11]

2.6.3.4 Τελεστές ενημέρωσης (Update Operators)

Εκτός από τους τελεστές ερωτημάτων η MongoDB διαθέτει και τελεστές ενημερώσεων. Κάποιοι από τους πιο γνωστούς και χρησιμοποιήσιμους είναι:

- **$inc :** Αύξηση της τιμής του πεδίου κατά συγκεκριμένο ποσό.
- **$mul :** Πολλαπλασιασμός της τιμής του πεδίου κατά συγκεκριμένο ποσό.
- **$rename :** Μετονομασία ενός πεδίου.
- **$set :** Ορισμός της τιμής ενός πεδίο σε ένα Έγγραφο.
- **$unset :** Διαγραφή πεδίου από ένα Έγγραφο.
- **$min :** Ενημέρωση του πεδίου εάν η καθορισμένη τιμή είναι μικρότερη από την υπάρχουσα.
- **$max :** Ενημέρωση του πεδίου εάν η καθορισμένη τιμή είναι μεγαλύτερη από την υπάρχουσα.
- **$currentDate :** Ορίζει την τιμή ενός πεδίου στην τρέχουσα ημερομηνία.
- **$addToSet :** Προσθήκη στοιχείων σε έναν πίνακα μόνο αν δεν υπάρχουν ήδη.
- **$pop :** Διαγραφή του πρώτου ή του τελευταίου στοιχείου ενός πίνακα.
- **$pull :** Διαγραφή όλων των στοιχείων ενός πίνακα που ταιριάζουν σε ένα συγκεκριμένο ερώτημα.

2.6.4 Διαγραφή (Delete)

2.6.4.1 Διαγραφή πολλών documents

Για να διαγράψουμε όλα τα documents ενός collection, περνάμε ένα άδειο φίλτρο document {} στη μέθοδο `db.collection.deleteMany()`.

Στο παράδειγμα που ακολουθεί, διαγράφονται όλα τα documents του inventory collection:

```
db.inventory.deleteMany({})
```

Η μέθοδος αυτή, επιστρέφει ένα document με το αποτέλεσμα της διαδικασίας που εκτελέστηκε.

Στον παρακάτω πίνακα, παραθέτουμε τις βασικές μεθόδους διαγραφής των εγγράφων ενός collection:

`db.collection.deleteOne()`	Διαγράφει το πολύ ένα document το οποίο πληροί τα φίλτρα που έχουν εφαρμοστεί, ακόμα και αν υπάρχουν πολλά documents που πληρούν τα κριτήρια αυτά.
`db.collection.deleteMany()`	Διαγράφει όλα τα documents που ταιριάζουν στα φίλτρα που έχουν εφαρμοστεί.
`db.collection.remove()`	Διαγράφει είτε ένα document, είτε πολλά, τα οποία ταιριάζουν στα φίλτρα που έχουν εφαρμοστεί

Πίνακας 4. Delete μέθοδοι της MongoDB [11]

2.7 Aggregation (Ομαδοποίηση)

Στη MongoDB, η λειτουργία του Aggregation [12] επεξεργάζεται εγγραφές δεδομένων και επιστρέφει τα αποτελέσματα των υπολογισμών που πραγματοποιήθηκαν. Οι τιμές από πολλαπλά documents ομαδοποιούνται και στην ομάδα που προκύπτει, εφαρμόζονται διάφοροι υπολογισμοί ώστε να προκύψει ένα αποτέλεσμα. Η αντίστοιχη λειτουργία του aggregation στην SQL είναι για παράδειγμα το count(*) σε συνδυασμό με το GROUP BY.

Η σύνταξη της aggregate() μεθόδου είναι η εξής:

```
db.COLLECTION_NAME.aggregate(AGGREGATE_OPERATION)
```

Παράδειγμα:
Έστω ότι τα παρακάτω δεδομένα περιέχονται σε ένα collection:

```
{
   _id: ObjectId(7df78ad8902c)
   title: 'MongoDB Overview',
   description: 'MongoDB is no sql database',
   by_user: 'tutorials point',
   url: 'http://www.tutorialspoint.com',
   tags: ['mongodb', 'database', 'NoSQL'],
   likes: 100
},
{
   _id: ObjectId(7df78ad8902d)
   title: 'NoSQL Overview',
   description: 'No sql database is very fast',
   by_user: 'tutorials point',
   url: 'http://www.tutorialspoint.com',
   tags: ['mongodb', 'database', 'NoSQL'],
   likes: 10
},
{
   _id: ObjectId(7df78ad8902e)
   title: 'Neo4j Overview',
   description: 'Neo4j is no sql database',
   by_user: 'Neo4j',
   url: 'http://www.neo4j.com',
   tags: ['neo4j', 'database', 'NoSQL'],
   likes: 750
},
```

Εάν από το παραπάνω collection θέλουμε να εμφανίσουμε μία λίστα με τον αριθμό των tutorials που έχουν γραφτεί από κάθε user, χρησιμοποιούμε την μέθοδο aggregate() ως εξής:

```
> db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial :
{$sum : 1}}}])
{ "_id" : "tutorials point", "num_tutorial" : 2 }
{ "_id" : "Neo4j", "num_tutorial" : 1 }
>
```

Το αντίστοιχο SQL query για το παραπάνω παράδειγμα είναι:

SELECT by_user, COUNT(*) FROM mycol GROUP BY by_user

2.8 MongoDB Replication

Το Replication [13] είναι η διαδικασία συγχρονισμού των δεδομένων μεταξύ πολλών servers. Το replication προσφέρει πλεονασμό, αυξάνει τη διαθεσιμότητα των δεδομένων με πολλαπλά αντίγραφα σε πολλούς διαφορετικούς servers και προστατεύει τη βάση δεδομένων από βλάβες που θα μπορούσαν να παρουσιαστούν εάν χαθεί ο server. Επιτρέπει επίσης την ανάκαμψη από σφάλματα υλικού και διακοπές στην παροχής των υπηρεσιών στον χρήστη. Με τη δημιουργία επιπλέον αντιγράφων των δεδομένων, δίνεται η δυνατότητα της χρήσης τους για ανάκαμψη από βλάβες, για backup ή για reporting.

2.8.1 Πώς γίνεται το Replication στη MongoDB

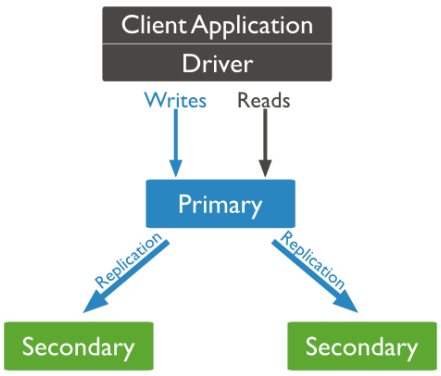
Η MongoDB, πραγματοποιεί replication χρησιμοποιώντας ένα replica set [13]. Ένα replica set, είναι μία ομάδα από mongod διεργασίες οι οποίες φιλοξενούν το ίδιο data set. Σε ένα replica (αντίγραφο), ένας κόμβος λειτουργεί ως πρωταρχικός και λαμβάνει τις λειτουργίες εγγραφής. Όλες οι υπόλοιπες διεργασίες, όπως οι δευτερεύουσες, εφαρμόζουν τις λειτουργίες του πρωταρχικού, ώστε να έχουν όλοι το ίδιο data set.

Κάθε ομάδα αντιγράφων (replica set) είναι μία ομάδα από δύο ή περισσότερους κόμβους η οποία μπορεί να διαθέτει μόνο έναν πρωταρχικό κόμβο. Απαιτούνται γενικότερα τουλάχιστον τρεις κόμβοι, ώστε ο ένας να λειτουργεί σαν πρωταρχικός και οι υπόλοιποι ως δευτερεύοντες. Έπειτα όλα τα δεδομένα αντιγράφονται από τον primary στους secondary κόμβους.

Σε περίπτωση που εφαρμοστεί αυτόματη λειτουργία failover ή κάποια συντήρηση, τότε εκλέγεται ένας νέος πρωταρχικός κόμβος.

Μετά την ανάκαμψη ενός κόμβου που έχει υποστεί βλάβη, μπορεί ο ίδιος να συμμετέχει και πάλι στο replica set λειτουργώντας ως δευτερεύον κόμβος.

Το παρακάτω διάγραμμα απεικονίζει τη λειτουργία του replication κατά την οποία η εφαρμογή πάντα αλληλεπιδρά με τον πρωταρχικό κόμβο, ο οποίος στη συνέχεια αντιγράφει τα δεδομένα στους δευτερεύοντες κόμβους.



Εικόνα 8. MongoDB Replication [13]

2.8.2 Δημιουργία ενός replica set

Για τη μετατροπή μίας MongoDB διεργασίας σε replica set, ακολουθούμε τα παρακάτω βήματα:

Απενεργοποιούμε τον MongoDB server που μπορεί να τρέχει εκείνη τη στιγμή

Ξεκινάμε ξανά τον MongoDB server, προσδιορίζοντας την επιλογή – replSet ως εξής:

```
mongod --port "PORT" --dbpath "YOUR_DB_DATA_PATH" --replSet "REPLICA_SET_INSTANCE_NAME"
```

Παράδειγμα:

```
mongod --port 27017 --dbpath "D:\set up\mongodb\data" --replSet rs0
```

- Ξεκινά ένα mongod instance με όνομα rs0, στην πόρτα 27017
- Συνδεόμαστε στο instance αυτό με την εκκίνηση μίας νέας γραμμής εντολών
- Στον Mongo client, γράφουμε την εντολή rs.initiate() για να δημιουργήσουμε ένα replica set
- Μπορούμε να δούμε τις ρυθμίσεις του replica set με το rs.conf() και να ελέγξουμε την κατάσταση του κάθε replica set με το rs.status()

2.8.3 Προσθήκη κόμβων σε ένα replica set

Για να προσθέσουμε επιπλέον μέλη σε μία ομάδα αντιγράφων, ξεκινάμε πολλαπλά instances σε διαφορετικά συστήματα. Μετά την εκκίνηση του mongo client, γράφουμε την εντολή rs.add(), η σύνταξη της οποίας φαίνεται παρακάτω:

```
>rs.add(HOST_NAME:PORT)
```

Παράδειγμα:

Έστω ότι έχουμε ένα mongod instance με όνομα mongod1.net το οποίο τρέχει στην πόρτα 27017. Για να προσθέσουμε το instance αυτό σε ένα replica set, διαμορφώνουμε την εντολή rs.add() ως εξής:

```
>rs.add("mongod1.net:27017")
```

Σε αυτό το σημείο, θα πρέπει να σημειώσουμε ότι μπορούμε να προσθέσουμε mongod instances σε ένα replica set, μόνο όταν έχουμε συνδεθεί στον primary node. Για να ελέγξουμε εάν έχουμε συνδεθεί σε έναν πρωταρχικό κόμβο, αρκεί να γράψουμε την εντολή db.isMaster() στον mongo client.

2.8.4 Πλεονεκτήματα χρήσης της μεθόδου replication

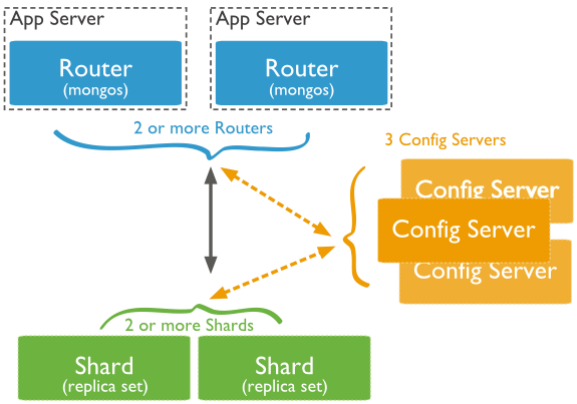
- Ασφαλή δεδομένα
- Υψηλή διαθεσιμότητα των δεδομένων (24/7)
- Ανάκαμψη από καταστροφές
- Δεν υπάρχει διακοπή στη λειτουργία κατά τη διάρκεια της συντήρησης
- Κλιμάκωση της λειτουργίας ανάγνωσης, καθώς υπάρχουν πολλαπλά αντίγραφα των δεδομένων στα οποία μπορεί να εφαρμοστεί η ανάγνωση
- Οι ομάδες αντιγράφων είναι ανεξάρτητες από την εφαρμογή

2.8.5 Χαρακτηριστικά των replica sets

- Είναι ένα σύμπλεγμα που αποτελείται από N κόμβους
- Κάθε κόμβος μπορεί να εκλεχθεί ως πρωταρχικός (primary)
- Όλες οι εγγραφές γίνονται στον πρωταρχικό κόμβο
- Αυτόματη διαδικασία failover
- Αυτόματη ανάκαμψη και ανάκτηση δεδομένων σε περίπτωση βλάβης

2.9 MongoDB Sharding

Το sharding [14] είναι μία διαδικασία αποθήκευσης δεδομένων σε πολλαπλές μηχανές, και είναι πρακτικά η λύση που παρέχει η MongoDB ώστε να πληρούνται οι απαιτήσεις της συνεχούς ανάπτυξης των δεδομένων (data growth). Καθώς το μέγεθος των δεδομένων αυξάνεται, μία μόνο μηχανή δεν επαρκεί για την αποθήκευση όλου του όγκου δεδομένων και μπορεί να μην παρέχει αποδεκτή ταχύτητα στις λειτουργίες ανάγνωσης και εγγραφής. Η μέθοδος του sharding, λύνει αυτά τα προβλήματα παρέχοντας οριζόντια κλιμάκωση (horizontal scaling).



Εικόνα 9. MongoDB Sharding [14]

Στο παραπάνω διάγραμμα, υπάρχουν τρία στοιχεία:

- Shards: Χρησιμοποιούνται για την αποθήκευση δεδομένων και παρέχουν υψηλή διαθεσιμότητα και συνοχή δεδομένων. Σε παραγωγικό περιβάλλον, κάθε shard είναι ένα ξεχωριστό replica set.
- Config Servers: Εδώ αποθηκεύονται τα μετα-δεδομένα του cluster. Τα δεδομένα αυτά περιέχουν μία αντιστοίχιση του συνόλου δεδομένων του cluster με τα αντίστοιχα shards. Το

query router χρησιμοποιεί αυτά τα metadata για να αναθέσει την κάθε λειτουργία σε συγκεκριμένα shards. Σε παραγωγικό περιβάλλον, τα sharded clusters έχουν ακριβώς τρεις config servers.

- Query Routers: Τα query routers είναι πρακτικά mongo διεργασίες, η οποίες αναλαμβάνουν τη διεπαφή με την εφαρμογή του client, κατευθύνουν την κάθε λειτουργία στο κατάλληλο shard και έπειτα επιστρέφουν τα αποτελέσματα στους clients. Ένα sharded cluster μπορεί να περιέχει περισσότερα από ένα query routers ώστε να μοιράζεται αποτελεσματικότερα ο φόρτος.

2.9.1 Πλεονεκτήματα χρήσης της μεθόδου sharding

- Όταν εφαρμόζεται replication, όλες οι λειτουργίες εγγραφής, γίνονται στο master node.
- Τα ευαίσθητα ερωτήματα γίνονται και αυτά στο master node.
- Ένα replica set, περιορίζεται στα 12 nodes
- Η μνήμη μπορεί να μην επαρκεί όταν η ενεργή βάση δεδομένων είναι μεγάλη σε μέγεθος
- Ο τοπικός δίσκος δεν διαθέτει επαρκή χώρο αποθήκευσης

2.10 Πότε δεν συνιστάται η χρήση της MongoDB

Παρόλο που η MongoDB είναι ευέλικτη και εύκολη στη χρήση, υπάρχουν ορισμένες περιπτώσεις στις οποίες δεν συνιστάται [15]:

- Όταν δεν υπάρχει πολλή διαθέσιμη μνήμη και μεγάλος αποθηκευτικός χώρος για τα δεδομένα. Το WiredTiger, έχει λύσει το μεγαλύτερο μέρος αυτού του προβλήματος, αλλά και πάλι η MongoDB δεν καθαρίζει αυτόματα τον χώρο στο δίσκο. Αυτό σημαίνει ότι όταν ένα document διαγράφεται, ο χώρος δεν απελευθερώνεται αυτόματα, αλλά χρειάζεται είτε restart είτε χειροκίνητο καθαρισμό.
- Όταν απαιτούνται πολλαπλές συσχετίσεις μεταξύ πινάκων
- Όταν πρέπει μία εφαρμογή να υποστηρίζει συναλλαγές. Για παράδειγμα, όταν πρέπει να ενημερωθεί παραπάνω από ένα document ή collection ανά user request, η χρήση της MongoDB μπορεί να κάνει corrupt τα δεδομένα, λόγω μη υποστήριξης ACID.

2.11 Θέματα ασφαλείας στη MongoDB

Όταν η MongoDB άρχισε να σχεδιάζεται, η ασφάλεια αυτής δεν ήταν το κύριο μέλημα των σχεδιαστών και για το λόγο αυτό υπάρχουν αρκετές «τρύπες» [16]. Στο κεφάλαιο αυτό παρουσιάζουμε τα κυριότερα θέματα ασφαλείας της MongoDB.

2.11.1 Αρχεία δεδομένων της MongoDB

Τα αρχεία δεδομένων είναι μη κρυπτογραφημένα και η Mongo δεν παρέχει κάποιες μεθόδους ή μηχανισμούς για την αυτόματη κρυπτογράφηση των αρχείων. Αυτό σημαίνει πως σε περίπτωση που κάποιος εισβάλει στο σύστημα αρχείων, μπορεί απευθείας να εξάγει οποιαδήποτε πληροφορία από τα αρχεία. Για την εξάληψη αυτού του θέματος, θα πρέπει η εφαρμογή να κρυπτογραφεί οποιαδήποτε ευαίσθητη πληροφορία πριν ακόμα αυτή γραφτεί στη βάση δεδομένων. Θα μπορούσαν επιπρόσθετα να εφαρμοστούν κανόνες πρόσβασης στο σύστημα αρχείων του εκάστοτε λειτουργικού συστήματος, ώστε να αποτραπεί η πρόσβαση μη εξουσιοδοτημένων χρηστών.

2.11.2 Διεπαφές χρηστών (Client Interfaces):

Η Mongo υποστηρίζει ένα δυαδικό, φυσικού επιπέδου πρωτόκολλο, χρησιμοποιώντας από προεπιλογή την TCP πόρτα 27017. Το πρωτόκολλο αυτό, χρησιμοποιείται από όλους του drivers και είναι ο πιο αποτελεσματικός τρόπος επικοινωνίας με τη Mongo. Εκτός από τους drivers της εφαρμογής, η βάση δεδομένων την Mongo χρησιμοποιεί την πόρτα αυτή για τη λειτουργία του replication. Επιπρόσθετα, εάν στον αριθμό πόρτας προσθέσουμε 1000, τότε προκύπτει ένας αριθμός πόρτας που μπορεί να χρησιμοποιηθεί σαν HTTP server (προεπιλεγμένη TCP πόρτα 28017). Ο συγκεκριμένος HTTP server, παρέχει ορισμένες στατιστικές μετρήσεις σε επίπεδο διαχείρισης, αλλά μπορεί επίσης να ρυθμιστεί κατάλληλα ώστε να παρέχει μία RESTful διεπαφή στη βάση δεδομένων, θέτοντας rest=true στο configuration αρχείο της βάσης δεδομένων ή χρησιμοποιώντας τη γραμμή εντολών. Το συγκεκριμένο πρωτόκολλο, δεν είναι ούτε κρυπτογραφημένο, ούτε συμπιεσμένο και ο εσωτερικός HTTP server δεν υποστηρίζει TLS ή SSL. Παρόλα αυτά, ο HTTP server μπορεί να «κρυφτεί» πίσω από ένα HTTP proxy server, όπως ο Apache HTTPD, ο οποίος προσφέρει ισχυρούς μηχανισμούς αυθεντικοποίησης και εξουσιοδότησης (authentication & authorization), όπως επίσης και SSL κρυπτογράφηση της σύνδεσης.

2.11.3 Πιθανότητες για μολυσματικές επιθέσεις

Η MongoDB χρησιμοποιεί την Javascript σαν εσωτερική scripting γλώσσα. Οι περισσότερες εντολές οι οποίες είναι διαθέσιμες προς τους προγραμματιστές, είναι ουσιαστικά μικρά javascript scripts. Είναι ακόμη πιθανή η αποθήκευση javascript functions μέσα στη βάση δεδομένων, στη συλλογή db.system.js, οι οποίες είναι έπειτα διαθέσιμες στους χρήστες της βάσης δεδομένων. Λόγω του ότι η Javascript είναι μία interpreted γλώσσα, οι πιθανότητες για injection επιθέσεις είναι αυξημένες. Για παράδειγμα, οι παρακάτω εκφράσεις, μπορούν να χρησιμοποιηθούν για να κάνουν το ίδιο query στη βάση δεδομένων με τη χρήση του where:

```
db.myCollection.find(
{ a : { $gt: 3 } }
);

db.myCollection.find(
{ $where: "this.a > 3" }
);

db.myCollection.find(
"this.a > 3"
);

db.myCollection.find(
{ $where: function() {
return this.a > 3;}
}
);
```

Στη δεύτερη και την τρίτη έκφραση, το where περνάται ως string το οποίο μπορεί να περιέχει τιμές η οποίες έχουν συνενωθεί απευθείας με τιμές που έχουν εισαχθεί από τον χρήστη. Στην τέταρτη έκφραση

το αντικείμενο $where είναι ένα Javascript function το οποίο θα χρησιμοποιηθεί από κάθε εγγραφή του collection. Δεν μπορεί να χρησιμοποιηθεί για να κάνει αλλαγές απευθείας στη βάση δεδομένων, εφόσον το $where function εκτελείται στη βάση με read-only δικαιώματα.

Παρόλα αυτά, αν κάποια εφαρμογή χρησιμοποιεί αυτού του τύπου where clauses, χωρίς να αποκλείει την πρόσβαση στα δεδομένα που έχουν εισαχθεί από τον χρήστη, τότε αυτά μπορούν να αντίστοιχα να είναι εκμεταλλεύσιμα και από μία injection επίθεση.

2.11.4 Αυθεντικοποίηση (Authentication)

Όταν η Mongo τρέχει σε Sharded mode, τότε δεν υποστηρίζει αυθεντικοποίηση. Παρόλα αυτά, όταν τρέχει σε stand-alone ή replica mode, υπάρχει δυνατότητα ενεργοποίησης του authentication. Η μόνη διαφορά μεταξύ της standalone και της replica λειτουργίας, είναι ότι στο replica-set mode, αυθεντικοποιούνται και οι χρήστες της βάσης, αλλά και κάθε replica server πρέπει να εξασφαλίσει την ταυτότητά του στους υπόλοιπους servers, πριν την είσοδό του σε ένα cluster. Και στις δύο λειτουργίες, η αυθεντικοποίηση βασίζεται σε ένα μυστικό κωδικό ο οποίος γνωστοποιείται από πριν σε όλους τους συμμετέχοντες.

Για το replica-set mode, η αυθεντικοποίηση από server σε server, γίνεται με το μυστικό κωδικό ο οποίος παρέχεται μέσω της keyfile παραμέτρου στο αρχείο ρυθμίσεων. Ο διαχειριστής του συστήματος, είναι υπεύθυνος να εξασφαλίσει ότι όλοι οι servers σε ένα replica-set cluster θα χρησιμοποιούν τον ίδιο μυστικό κωδικό.

Για τις συνδέσεις των χρηστών, κάθε χρήστης ρυθμίζεται κατά τη σύνδεση στην αντίστοιχη βάση δεδομένων (σε replica-set βάσεις δεδομένων, ο master server πρέπει να έχει ενημερωθεί) καλώντας τη μέθοδο db.addUser(). Ο διαχειριστής (admin) της βάσης δεδομένων, προστίθεται στη βάση ως DBA user με τα ειδικά δικαιώματα που απαιτούνται για την διαχείριση των συστημάτων.

Ο κωδικός πρόσβασης αποθηκεύεται με τη μορφή ενός MD5 hash του string <username>:mongo:<password>, και μπορεί εύκολα να διαβαστεί από τα αρχεία δεδομένων του διαχειριστή. Αυτό σημαίνει ότι οποιοσδήποτε έχει πρόσβαση στα αρχεία δεδομένων, μπορεί εύκολα να ανακτήσει τους κωδικούς πρόσβασης όλων των χρηστών που έχουν δημιουργηθεί στη βάση δεδομένων.

Ο αλγόριθμος MD5 δεν ανήκει στους πιο ασφαλείς αλγορίθμους, και είναι σχεδόν αυτονόητο ότι όποιος θέλει να επιτεθεί στο σύστημα, μπορεί να κάνει μία brute force επίθεση μέσω του wire-level πρωτοκόλλου, το οποίο θα επιφέρει εύκολα τα επιθυμητά αποτελέσματα. Εάν η εφαρμογή χρησιμοποιεί το RESTful API, τότε ο εσωτερικός HTTP server της Mongo, κρύβεται πίσω από έναν proxy, ο οποίος μπορεί να χρησιμοποιηθεί αποτελεσματικά και για authentication όπως και για authorization των χρηστών αλλά και των διαφόρων άλλων οντοτήτων της βάσης δεδομένων.

2.11.5 Εξουσιοδότηση (Authorization)

Όταν η MongoDB τρέχει σε Sharded mode, δεν παρέχει ούτε authentication, και κατ' επέκταση ούτε authorization. Εάν έχει ενεργοποιηθεί η λειτουργία αυθεντικοποίησης, τότε η βάση δεδομένων της Mongo υποστηρίζει δύο τύπους χρηστών: αυτούς που έχουν μόνο δικαίωμα ανάγνωσης (read-only) και αυτούς που έχουν δικαίωμα ανάγνωσης και εγγραφής (read & write).

Οι χρήστες που έχουν δικαιώματα μόνο για ανάγνωση, μπορούν να τρέξουν queries στη βάση δεδομένων στην οποία έχουν εξουσιοδοτηθεί και να δουν τα αποτελέσματα αυτών, ενώ οι read & write χρήστες έχουν πλήρη πρόσβαση σε όλα τα δεδομένα της βάσης. Κάθε χρήστης που έχει οριστεί στην admin βάση δεδομένων, θεωρείται ως DBA user. Τέτοιου είδους χρήστες, έχουν πλήρη πρόσβαση για

ανάγνωση και εγγραφή σε όλες τις βάσεις δεδομένων που ανήκουν σε ένα cluster. Παρόλο που η Mongo δεν υποστηρίζει απευθείας κάτι τέτοιο, όταν χρησιμοποιείται το RESTful API, όπως αναφέραμε και παραπάνω, οι proxy servers είναι αυτοί που παρέχουν πιο ισχυρή αυθεντικοποίηση. Ένας reverse proxy, μπορεί να προσφέρει διαφορετικά δικαιώματα για το test namespace σε σχέση με το data namespace.

2.11.6 Auditing (Έλεγχος)

Η MongoDB δεν προσφέρει μηχανισμούς για έλεγχο των δραστηριοτήτων που διενεργούνται μέσα στη βάση δεδομένων. Όταν δημιουργείται μία νέα βάση δεδομένων, τότε η Mongo δημιουργεί μία νέα εγγραφή στα αρχεία καταγραφής για τη δημιουργία της βάσης. Παρόλα αυτά, έπειτα από την κατανομή των αρχείο δεδομένων, στα log files δεν εγγράφεται τίποτα παραπάνω για περαιτέρω εισαγωγές, ενημερώσεις ή ερωτήματα (queries).

2.12 Επίλογος

Αφού αναλύσαμε και παρουσιάσαμε τις δυνατότητες της MongoDB, στο επόμενο κεφάλαιο παρουσιάζονται οι ιδιότητες των Συστημάτων Διαχείρισης Περιεχομένου (CMSs)

Κεφάλαιο 3ο: CMS (Content Management System)

3.1 Εισαγωγή

Ο ρυθμός ανανέωσης της πληροφορίας μεταβάλλεται συνεχώς από τις αρχές του 21ου αιώνα. Πλοηγός της νέας κοινωνίας της πληροφορίας αποτελεί το Διαδίκτυο, που παρουσιάζει ολοένα και μεγαλύτερη διεισδυτικότητα στα σύγχρονα νοικοκυριά. Οι χιλιάδες χρήστες πλέον απαιτούν «φρέσκες» πληροφορίες, με ανανέωση τουλάχιστον ανά ώρα ή και ανά λεπτά, για παράδειγμα στην περίπτωση παρακολούθησης του χρηματιστηρίου. Η απαίτηση αυτή συνοδεύεται ταυτόχρονα από την ανάγκη για την ύπαρξη μία ευέλικτης πλατφόρμας για την παρουσίαση αυτών των πληροφοριών. Τα δύο αυτά χαρακτηριστικά, σύμφυτα της ανάπτυξης του Διαδικτύου, επηρεάζουν μία μεγάλη γκάμα οργανισμών, όχι απαραίτητα κερδοσκοπικών. Για παράδειγμα, μία ηλεκτρονική εφημερίδα χρειάζεται εξίσου το κοινό της, όπως και μία εμπορική επιχείρηση, για να επιβιώσει αρχικά και για να μπορεί να ασκεί επιρροή στην σύγχρονη πραγματικότητα αφετέρου.

Η αυτοματοποίηση των διαδικασιών δημιουργίας των πληροφοριών, που αποτελούν το περιεχόμενο του Διαδικτύου, δημοσίευσης τους και παρουσίασης τους συνιστά το επόμενο βήμα στις προηγούμενες απαιτήσεις. Ο μεγάλος όγκος της πληροφορίας σε συνδυασμό με την απαιτούμενη τεχνική γνώση δεν επέτρεπε στους οργανισμούς να επιτύχουν την ισορροπία ανάμεσα σε ένα εύχρηστο και ελκυστικό περιβάλλον παρουσίασης και σε ένα συνεχώς ανανεώσιμο περιεχόμενο, που θα τους εξασφάλιζε μία σταθερή βάση επισκεψιμότητας στη ιστοσελίδα τους. Όταν δε έμπαινε και ο παράγοντας του ελέγχου της ροής της πληροφορίας από πολλαπλά άτομα, η κατάσταση γινόταν ακόμη πιο δύσκολη. Αποτέλεσμα ήταν η δημιουργία μεγάλων ιστοχώρων με καλή σχεδίαση, αλλά ξεπερασμένο χρονικά περιεχόμενο, ή με κακή σχεδίαση χωρίς μεγάλα περιθώρια ευελιξίας, αλλά με υπέρ-ανανεωμένο περιεχόμενο.

Η έλλειψη τεχνικών γνώσεων από τα στελέχη του οργανισμού οδηγούσε τις επιχειρήσεις σε δημιουργία γραφείων ή σε εκμίσθωση ειδικευμένων εταιριών για την διατήρηση των ιστοσελίδων τους. Εκτός από το φανερό κόστος της κίνησης αυτής, η λύση της δημιουργίας ενός ειδικού γραφείου παρουσίαζε σημαντικά προβλήματα. Λίγα άτομα με τεχνικές γνώσεις επιμερίζονταν τον τεράστιο όγκο των πληροφοριών του ιστοχώρου, ενώ επιμερίζονταν ταυτόχρονα και όλες τις λειτουργίες, από την εύρεση του περιεχομένου, την επεξεργασία του, την δημοσίευση του και την αποθήκευση του για μελλοντική χρήση. Συνέπεια ήταν να μην μπορεί το γραφείο πολλές φορές να διαχειριστεί τον τεράστιο όγκο των πληροφοριών, αυτές να δημοσιεύονται με καθυστέρηση και να μην υπάρχει πολυφωνία και πλούτος περιεχομένου. Ιδιαίτερα, αν η ιστοσελίδα ήταν μεγάλη, τότε πολλές φορές το περιεχόμενο της διαμοιράζονταν σε πολλά γραφεία, με αποτέλεσμα έναν ιστοχώρο με έλλειψη διασύνδεσης και χωρίς πολλές φορές καμία συνοχή.

Το τοπίο λοιπόν ήταν γόνιμο για την δημιουργία των ηλεκτρονικών εργαλείων, που θα έδιναν λύση στο πρόβλημα της επιτυχημένης ηλεκτρονικής παρουσίας των οργανισμών στο διαδίκτυο. Τα CMS επιτρέπουν στους οργανισμούς να δημιουργούν, αλλά και να εισάγουν έτοιμο πολυμεσικό υλικό. Να πιστοποιούν τους χρήστες του συστήματος και να επιμερίζουν ξεχωριστούς ρόλους στον καθένα στον κύκλο της λειτουργίας τους. Επίσης, επιτρέπουν τον προσδιορισμό εργασιών ροής του περιεχομένου, συχνά σε συνδυασμό με την λειτουργία των ειδοποιήσεων συμβάντων, που επιτρέπουν στους διαχειριστές του περιεχομένου να ειδοποιούνται για οποιαδήποτε αλλαγή.

Τα CMS δίνουν ακόμη στους χρήστες την δυνατότητα να εντοπίζουν και να διαχειρίζονται πολλαπλές εκδόσεις ενός μόνο αρχείου περιεχομένου, να το αποθηκεύουν σε μία ξεχωριστή βάση δεδομένων, ενώ ταυτόχρονα προσφέρουν την δυνατότητα ευρετηρίου, διευκολύνοντας τον έλεγχο και την επαναφορά παλαιότερου υλικού της ιστοσελίδας. Το βασικότερο, όμως, χαρακτηριστικό που προσφέρουν είναι η δυνατότητα διαχωρισμού του περιεχομένου από την παρουσίαση της ιστοσελίδας.

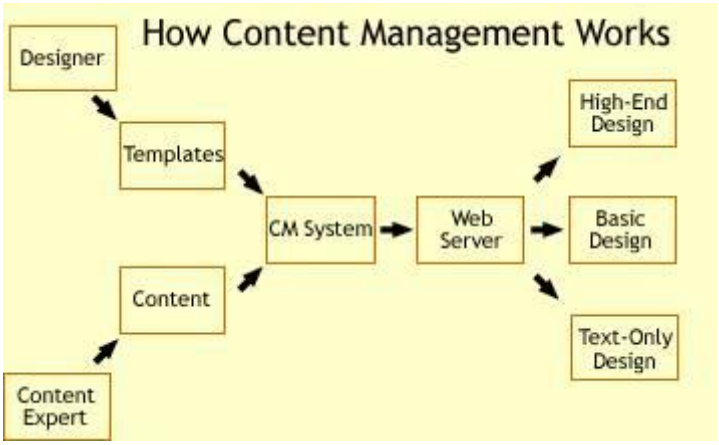
Όλα αυτά τα χαρακτηριστικά θα αναλυθούν διεξοδικά στην συνέχεια παρέχοντας το θεωρητικό υπόβαθρο της παρούσας εργασίας. Ταυτόχρονα, θα αναλυθεί: ο τρόπος λειτουργίας των CMS, με ποιον τρόπο θα πρέπει να γίνεται η επιλογή του κατάλληλου CMS, καθώς επίσης και ποια είναι τα πλεονεκτήματα και μειονεκτήματα τους.

3.2 Ορισμός- Περιγραφή

Το Content Management System (CMS) είναι μία μορφή λογισμικού για ηλεκτρονικούς υπολογιστές, που αυτοματοποιεί τις διαδικασίες δημιουργίας, οργάνωσης, ελέγχου και δημοσίευσης περιεχομένου σε μία πληθώρα μορφών. Τα περισσότερα CMS έχουν την δυνατότητα να διαχειριστούν περιεχόμενο στις εξής μορφές: κείμενα, εικόνες, βίντεο, java animation, πρότυπα σχεδίασης, βάσεις δεδομένων κ.α. Πολλές φορές ένα CMS επιτρέπει και την ομαδική δημιουργία κειμένων και άλλου υλικού, για αυτό συχνά χρησιμοποιείται, για παράδειγμα, στα εκπαιδευτικά προγράμματα πολλών εταιριών. Τα CMS χρησιμοποιούνται συχνά και για την αποθήκευση, τον έλεγχο, την διαχείριση και την δημοσίευση εκδόσεων, ο προσανατολισμός των οποίων εξαρτάται από τον φορέα, στον οποίο ανήκει το περιεχόμενο. Έτσι, μπορεί οι εκδόσεις αυτές να αποτελούνται από ειδησεογραφικά άρθρα, εγχειρίδια λειτουργίας, τεχνικά εγχειρίδια, οδηγίες πωλήσεων έως και εμπορικό διαφημιστικό υλικό.

Ένα Web Content Management System ή Web Publishing System είναι η μορφή λογισμικού, που παρέχει επιπρόσθετες δυνατότητες, για την διευκόλυνση των απαραίτητων εργασιών δημοσίευσης ηλεκτρονικού περιεχομένου σε μία ιστοσελίδα . Τα Web CMS έχουν την μεγαλύτερη διείσδυση στους οργανισμούς σήμερα, για αυτό και θα αποτελέσουν τον κορμό της παρούσας εργασίας. Αποτελούν ένα συνδυασμό μία μεγάλης βάσης δεδομένων, ενός συστήματος αρχειοθέτησης και άλλων στοιχείων λογισμικού, τα οποία χρησιμοποιούνται για την αποθήκευση και την μετέπειτα ανάκτηση των δεδομένων, καθώς επίσης χρησιμοποιούνται και για τις διάφορες ξεχωριστές λειτουργίες των CMS. Αυτά τα συστήματα ηλεκτρονικής δημοσίευσης, λοιπόν, γίνεται φανερό ότι διαφέρουν από τις απλές βάσεις δεδομένων υπό την έννοια ότι μπορούν να καταχωρήσουν κείμενο, ηχητικά αποσπάσματα, αποσπάσματα βίντεο ή εικόνες.

Οι χρήστες των Web CMS μπορούν να εντοπίσουν σχετικό υλικό στην βάση δεδομένων, ψάχνοντας με κριτήριο μία λέξη-κλειδί, τον συγγραφέα του κειμένου, την ημερομηνία δημιουργίας του αρχείου κτλ. Έτσι, μπορούν να αποτελέσουν πλέον τα Web CMS μία πύλη πληροφοριών, ή οποία μπορεί να χρησιμεύσει σαν ραχοκοκαλιά για την διαχείριση δεδομένων του ιδιοκτήτη της ιστοσελίδας. Για παράδειγμα, θα μπορούσε να χρησιμεύσει αποθηκεύοντας κάθε άρθρο, που δημοσιεύτηκε σε μία ηλεκτρονική εφημερίδα τα τελευταία τρία χρόνια, και δημιουργώντας ένα ευρετήριο. Έτσι δημιουργεί στην ουσία αυτόματα ένα αρχείο της εφημερίδας εύχρηστο και προσβάσιμο σε κάθε συντάκτη, χωρίς να χρειάζεται να εκτυπώνεται κάθε άρθρο και να διατηρείται ένα ογκώδες και απροσπέλαστο αρχείο.



Εικόνα 10. Η βάση της λειτουργίας όλων των CMS [17]

Ταυτόχρονα, πέρα από τις δυνατότητες σχετικά με την διαχείριση βάσεων δεδομένων, τα λογισμικά αυτά επιτρέπουν στον καθένα να συνεισφέρει πληροφορίες σε μία ιστοσελίδα με την χρήση μίας Γραφικής Διασύνδεσης Χρήστη (Graphical User Interface- GUI). Η διασύνδεση αυτή βασίζεται σε προκατασκευασμένα πρότυπα της ιστοσελίδας και παρέχει μία πλατφόρμα για την εισαγωγή δεδομένων σε κάθε τμήμα της ιστοσελίδας αυτής, χωρίς να είναι απαραίτητη η γνώση εξειδικευμένων γλωσσών προγραμματισμού. Επομένως, μπορούν πλέον οι συντάκτες των ιστοσελίδων να διαχωριστούν από τους τεχνικούς και να εισάγουν απευθείας δεδομένα. Διαχωρίζεται δηλαδή το περιεχόμενο από την παρουσίαση της ιστοσελίδας, που αποτελεί ένα από τα βασικά πλεονεκτήματα των CMS.

Τα Web CMS μπορούν ακόμη να διανείμουν υλικό σε πελάτες και εταιρικούς συνεργάτες πέρα από τα όρια ενός οργανισμού, παραδείγματος χάρη με την αυτόματη αποστολή newsletters στους πελάτες και την αυτόματη σύνταξη Δελτίων Τύπου και στατιστικών στοιχείων και την ηλεκτρονική αποστολή τους σε συνεργαζόμενες επιχειρήσεις ή ΜΜΕ. Ο πυρήνας, όμως, ενός CMS είναι η διαχείριση του περιεχομένου σε όλο τον κύκλο ζωής της πληροφορίας, δηλαδή από την παραγωγή της μέχρι την δημοσίευση της, αλλά και την μετέπειτα αποθήκευση της.

Τα CMS, επομένως, είναι όλα βασισμένα στην ίδια ιδέα (σχήμα 3.1): η διαχείριση περιεχομένου επιτρέπει στους σχεδιαστές να επικεντρωθούν στην σχεδίαση με το χτίσιμο προτύπων (templates). Από την άλλη, οι συντάκτες χτίζουν το περιεχόμενο σε ξεχωριστό περιβάλλον, ο κεντρικός διακομιστής παίρνει το περιεχόμενο, το εισάγει στο σωστό template και το στέλνει όλο μαζί, καθαρά περιτυλιγμένο, στους τελικούς χρήστες.

3.3 Πλεονεκτήματα

Υπάρχουν πολλά πλεονεκτήματα, που απορρέουν από την χρήση ενός CMS. Μερικά από αυτά είναι γενικά και μερικά εξαρτώνται από τα ιδιαίτερα χαρακτηριστικά του συγκεκριμένου CMS, που χρησιμοποιείται.

3.3.1 Γενικά Πλεονεκτήματα

Στα γενικά πλεονεκτήματα θα μπορούσαμε να εντάξουμε την μείωση των εξόδων για την διατήρηση μίας ιστοσελίδας και την αύξηση του εισοδήματος χάριν στην επιτυχημένη παρουσία της ιστοσελίδας αυτής. Ακόμη, σημαντικό πλεονέκτημα είναι η δυνατότητα ιεράρχησης και ροής της διαδικασίας δημιουργίας και δημοσίευσης αντικειμένων στην ιστοσελίδα με την χρήση των CMS. Συνεπώς, ιδιαίτερα στην δημιουργία ενός ιστοχώρου, όπου πολλά άτομα θα έχουν πρόσβαση, ώστε να εισάγουν

υλικό και να διατηρήσουν ενημερωμένη την ιστοσελίδα, χρειάζεται ένας έλεγχος των σταδίων, που θα ακολουθήσει η πληροφορία για να δημοσιευτεί. Ακόμη, με την βοήθεια των CMS μπορεί να αυξηθεί κατακόρυφα η ποιότητα μίας ιστοσελίδας με την χρήση υψηλής ποιότητας προτύπων σχεδίασης, που θα δίνουν μία εντυπωσιακή εικόνα για τον ιδιοκτήτη τους. Επίσης, τα πρότυπα αυτά μπορούν και να προσδίδουν την ταυτότητα και τον χαρακτήρα του, όπως συμβαίνει για παράδειγμα με την χρήση του χαρακτηριστικού κόκκινου χρώματος στην ιστοσελίδα της Coca-Cola.

Στα γενικά πλεονεκτήματα ενός Συστήματος Διαχείρισης Περιεχομένου, θα μπορούσαν να ενταχθούν και οι λιγότερες ανάγκες εκπαίδευσης, που απαιτεί. Με τις έτοιμες φόρμες εισαγωγής, μορφοποίησης και προεπισκόπησης, που προσφέρουν, δεν απαιτούνται πλέον ειδικές γνώσεις προγραμματισμού και σχεδίασης ιστοσελίδων. Με απλές γνώσεις χρήσης ηλεκτρονικών υπολογιστών, που είναι πλέον απαραίτητες στους εργαζόμενους κάθε τομέα της παραγωγής, αλλά και γενικότερα στην καθημερινή ζωή, μπορεί κάποιος να δημιουργήσει ένα εντυπωσιακό, περιεκτικό και ενημερωμένο ιστοχώρο. Σύμφωνα με την προηγούμενη διατύπωση, μία από τις βασικές συνέπειες των CMS θα είναι η μείωση του τεχνικού τμήματος, που απαιτείται για την διαχείριση μίας ιστοσελίδας σε ένα μικρό αριθμό τεχνικών, που θα χρειάζονται για την σωστή λειτουργία και συντήρηση των CMS. Επομένως, αυξάνεται ακόμη περισσότερο το κέρδος από την λειτουργία τους.

3.3.2 Ειδικά Πλεονεκτήματα

Πιο εξειδικευμένα πλεονεκτήματα από την χρήση ενός Content Management System μπορεί να έχει ένας οργανισμός βραχυπρόθεσμα και ανάλογα με το είδος του CMS, που χρησιμοποιεί. Πρώτον, ένας οργανισμός μπορεί να αποκεντρώσει την διατήρηση του περιεχομένου της ιστοσελίδας του, μειώνοντας τις οποιεσδήποτε καθυστερήσεις. Πλέον τα βήματα, που ακολουθούνται, μειώνονται και απλουστεύονται, ενώ η δημιουργία του περιεχομένου μπορεί να διανεμηθεί σε πολλούς. Εξαιτίας της ομοιομορφίας του προτύπου σχεδίασης που προσφέρει το κάθε CMS, πλέον μπορούν να δημιουργηθούν συνεκτικοί, αλλά και πολύ πλούσιοι σε περιεχόμενο ιστοχώροι, αποτέλεσμα της εργασίας πολλών διαφορετικών ανθρώπων και όχι λίγων τεχνικά καταρτισμένων.

Σε ένα δεύτερο επίπεδο, ο διαχωρισμός της λειτουργικότητας και της παρουσίασης της ιστοσελίδας από την δημοσίευση και το περιεχόμενο αντίστοιχα, μπορεί να συνεισφέρει σημαντικά σε έναν οργανισμό. Μπορεί να βοηθήσει στην καλύτερη ιεράρχηση των υπεύθυνων για την δημιουργία και διαχείριση της ιστοσελίδας, καθώς επίσης και στην εστίαση του κάθε υπεύθυνου συγκεκριμένα σε κάποιους τομείς της ιστοσελίδας. Αποτέλεσμα είναι ο καλύτερος καταμερισμός της εργασίας, ώστε να προκύψουν τα μέγιστα δυνατά αποτελέσματα. Για παράδειγμα, ο διευθυντής πωλήσεων σε μία επιχείρηση μπορεί να έχει την δική του ενότητα στην ιστοσελίδα της επιχείρησης, όπου δημοσιεύει τους ισολογισμούς, τους προϋπολογισμούς και τα μελλοντικά επιχειρηματικά σχέδια της επιχείρησης. Από την άλλη, ο υπεύθυνος τύπου της επιχείρησης έχει στην διάθεση του επίσης την δική του ενότητα, ώστε να δημοσιεύει Δελτία Τύπου, ειδήσεις σχετικά με την επιχείρηση, νέες καμπάνιες διαφημιστικές κ.τ.λ.

Αμφότερες αυτές οι ενότητες παρουσιάζουν μία ομοιομορφία, χωρίς να είναι φανερή η διαφορετική ταυτότητα του υπεύθυνου, ενώ στην πρώτη σελίδα υπάρχουν σύντομες καταχωρήσεις με υπερσυνδέσεις προς όλα όσα καταχωρούνται εσωτερικά. Παραδείγματος χάρη, μπορεί να υπάρχει ένα ημερολόγιο στο οποίο μπορεί να προστίθεται αυτόματα η καταχώρηση ενός γεγονότος, όταν αυτό καταχωρείται στην ενότητα του γραφείου τύπου. Έτσι, ένας οργανισμός μπορεί να χρησιμοποιήσει τα καλύτερα στελέχη του για κάθε τμήμα της ιστοσελίδας του, χωρίς να κινδυνεύει η εικόνα της ιστοσελίδας και η λειτουργικότητα της.

Εμφανή είναι και τα οφέλη από την παροχή πληροφοριών σε τακτά χρονικά διαστήματα, βασική δυνατότητα που προσφέρουν τα CMS στους οργανισμούς. Αποτέλεσμα είναι να αυξάνονται θεαματικά οι επισκέπτες, που επισκέπτονται την ιστοσελίδα ή επιστρέφουν σε αυτή, καθώς μέχρι τώρα δεν μπορούσαν να εντοπίσουν τις άμεσες πληροφορίες, που χρειάζονταν. Επιπλέον, εκτός από την αύξηση των επισκεπτών, αυξάνεται και η συχνότητα επισκεψιμότητας της ιστοσελίδας, αφού πλέον ο ίδιος επισκέπτης την επισκέπτεται συχνότερα, για να μπορέσει να βρει νέες πληροφορίες. Σε αυτήν την περίπτωση, ο κερδοσκοπικός οργανισμός κερδίζει από την δημιουργία πολλές φορές ενός μεγάλου πελατολογίου μέσω του Διαδικτύου, ενώ ο μη κερδοσκοπικός οργανισμός από την αύξηση της επιρροής του. Βασικό στοιχείο εδώ είναι ότι σχεδόν όλα τα CMS παρέχουν στατιστικά στοιχεία σχετικά με την επισκεψιμότητα, την συχνότητα επισκεψιμότητας και τις επιλογές των επισκεπτών της ιστοσελίδας.

Επιπροσθέτως, ένα από τα πιο σημαντικά πλεονεκτήματα αποτελεί η δυνατότητα πολλαπλών δημοσιεύσεων της πληροφορίας σε διάφορα κανάλια. Ως εκ τούτου μπορεί ένας οργανισμός να δημοσιεύσει αυτόματα περιεχόμενο σε διάφορα σημεία στην κεντρική σελίδα του, σε διάφορα τμήματα του δικτυακού τόπου του, αλλά πλέον μπορεί πολύ γρήγορα και αυτόματα να δημοσιεύσει υλικό και σε διάφορες συνεργαζόμενες ιστοσελίδες άλλων οργανισμών. Για παράδειγμα, ένα υποκατάστημα της Coca-Cola στην Ελλάδα βγάζει στον ισολογισμό του υψηλά κέρδη. Ο ισολογισμός δημοσιεύεται στην ιστοσελίδα την εταιρίας στην Ελλάδα και γράφεται ένα μικρό άρθρο σχετικά με τις αιτίες, που οδήγησαν στην υψηλή αυτή κερδοφορία. Ταυτόχρονα, όμως, το νέο αυτό δημοσιεύεται στην διεθνή ιστοσελίδα της Coca-Cola στο τμήμα των νέων και παρέχεται μία υπερσύνδεση προς την ελληνική σελίδα της εταιρίας και προς το συγκεκριμένο άρθρο. Η παγκοσμιοποίηση του 21ου αιώνα και του Διαδικτύου μπορεί να οδηγήσει λόγω του προηγούμενου παραδείγματος σε υψηλές επενδύσεις στις μετοχές τις εταιρίες στην Ελλάδα μετά από την ανάγνωση αυτού του άρθρου διεθνώς.

Η παροχή προσωποποιημένων υπηρεσιών αποτελεί την κορωνίδα στις υπηρεσίες, που προσφέρουν τα CMS, καθώς στον σύγχρονο εξατομικευμένο κόσμο το νέο μοντέλο της πληροφόρησης βασίζεται πλέον στις επιθυμίες και τα ιδιαίτερα χαρακτηριστικά του κάθε ατόμου. Είναι απαραίτητο πια στην ιστοσελίδα κάθε μεγάλου οργανισμού να υπάρχει μία περιοχή μελών, είτε αυτοί αποτελούν τα μέλη του οργανισμού, είτε αποτελούν τους επισκέπτες της ιστοσελίδας, που επιθυμούν να ενημερώνονται σχετικά με τον οργανισμό, να έχουν οικονομικές σχέσεις μαζί του ή να συμμετάσχουν στην βελτιστοποίηση του. Τα CMS παρέχουν την δυνατότητα για την δημιουργία τέτοιων υπηρεσιών, συμβάλλοντας στην παροχή καλύτερων υπηρεσιών του οργανισμού προς τους επισκέπτες της ιστοσελίδας του και μεγαλύτερη ικανοποίηση από τα μέλη του είτε εσωτερικά είτε εξωτερικά.

Τέλος, πολύ σημαντικό πλεονέκτημα, που θα έπρεπε να αναπτυχθεί διεξοδικότερα, είναι το ζήτημα του κόστους, που θα εξοικονομήσει ένας οργανισμός από την χρήση των CMS. Ήδη έχει αναφερθεί η μείωση του κόστους, εξαιτίας της μείωσης του τεχνικού προσωπικού, που απαιτείται για την διαχείριση της ιστοσελίδας. Η εξοικονόμηση, όμως για την εταιρία δεν περιορίζεται μόνο σε αυτόν τον τομέα. Επιπροσθέτως, θα μειωθεί το κόστος για την δημιουργία του εταιρικού branding μιας επιχείρησης και των εξόδων μάρκετινγκ, όταν πρόκειται για έναν κερδοσκοπικό οργανισμό, ή των εξόδων διαφήμισης και πρόσβασης σε ενδιαφερόμενους για την πληροφόρηση τους, όταν πρόκειται για μη κερδοσκοπικό οργανισμό. Τα CMS θα βελτιώσουν την παραγωγικότητα του εργατικού δυναμικού του οργανισμού, που σχετίζεται με την διαχείριση της ιστοσελίδας και θα μειώσουν κατακόρυφα τις τεχνικές γνώσεις, που απαιτούνται γενικότερα για την διαχείριση των ιστοσελίδων. Επομένως, θα μειωθούν τα έξοδα για την εκπαίδευση των μελών του οργανισμού και θα αυξηθούν τα οφέλη.

3.4 Χαρακτηριστικά

Τα χαρακτηριστικά των CMS αφορούν στη σύνθεση κάθε τέτοιου τύπου λογισμικού. Υπάρχουν πολλαπλά χαρακτηριστικά στα πιο απλά πακέτα, ενώ έχουν αναδειχτεί ακόμη και πιο πλούσιες σε χαρακτηριστικά λύσεις. Σημαντικό είναι σε αυτό το σημείο να τονιστεί, ότι τα open-source CMS, των οποίων ο κώδικας προγραμματισμού διατίθεται δωρεάν στο Διαδίκτυο, έχουν θεωρητικά άπειρες δυνατότητες βελτίωσης, σε σχέση με τα εμπορικά (commercial), τα οποία έχουν κάποιο κόστος και η βελτίωση των χαρακτηριστικών τους μπορεί να γίνει μόνο από την ίδια την εταιρεία δημιουργίας τους. [17]

3.5 Βασικά Χαρακτηριστικά

Βάση Δεδομένων Περιεχομένου: πρόκειται για μία βάση δεδομένων, η οποία συγκεντρώνει και ιεραρχεί όλο το περιεχόμενο, το οποίο πρόκειται να δημοσιευτεί στην ιστοσελίδα. Οι λύσεις των Συστημάτων Διαχείρισης Περιεχομένου έχουν, όπως είναι φυσικό, την δυνατότητα να διαχειρίζονται μία πολύ μεγάλη ποικιλία περιεχομένου, καθώς επίσης και διάφορες μορφές του περιεχομένου αυτού. Μπορούν να διαχειρίζονται διάφορες μορφές κειμένων, αρχείων (PDF, Word, Excel, PowerPoint, Zip), άρθρα, Δελτία Τύπου, εικόνες, streaming ήχου και βίντεο, html, γραφικά, υπερσυνδέσεις κ.α.

Βάση Δεδομένων Ατόμων: πρόκειται για μία βάση δεδομένων όλων των ατόμων που σχετίζονται με την ιστοσελίδα, την οποία διαχειρίζεται το CMS. Αυτά τα άτομα μπορεί να είναι επισκέπτες, μέλη, εγγεγραμμένοι στα newsletters της ιστοσελίδας, εθελοντές κ.λ.π. Σημειώνεται εδώ ότι δεν παρέχουν όλα τα CMS αυτήν την δυνατότητα, καθώς πολλές φορές βασίζονται στην βάση δεδομένων της Εξυπηρέτησης πελατών, με την οποία πολλές φορές τα CMS μπορούν να συνεργαστούν.

Βάση Διαχείρισης Χρηστών: πρόκειται για μία βάση δεδομένων, που αποτελείται από τα στοιχεία όλων των διαχειριστών και των συντακτών περιεχομένου της ιστοσελίδας, που διαχειρίζεται το CMS. Σε αυτήν την βάση αποθηκεύονται οι κωδικοί των χρηστών αυτών, καθώς επίσης οι συσχετισμένοι ρόλοι τους και τα καθήκοντα τους.

Πληροφοριακή Αρχιτεκτονική (Information Architecture- IA): πρόκειται για τον χάρτη πλοήγησης της ιστοσελίδας. Ένα CMS επιτρέπει στον διαχειριστή του να εγκαταστήσει και να διαχειριστεί την Πληροφοριακή Αρχιτεκτονική και να ρυθμίσει την παρουσίαση των σχετικών μενού πλοήγησης.

Σχεδίαση Παρουσίασης: πρόκειται για την οπτική και την αίσθηση της ιστοσελίδας, όπως αυτές δημιουργούνται μέσω της γραφικής σχεδίασης της. Η παρουσίαση της ιστοσελίδας χωρίζεται σε δύο μέρη: **Α) Πλαίσιο:** αναφέρεται στην εμφάνιση των δομικών χαρακτηριστικών της σελίδας, όπως για παράδειγμα της κεφαλίδας, του υποσέλιδου, της αριστερής, κεντρικής και δεξιάς στήλης, καθώς επίσης και των κύριων στοιχείων πλοήγησης. **Β) Γραφική Προσέγγιση:** αναφέρεται στην χρωματική παλέτα, τα είδη και τα μεγέθη των γραμματοσειρών και τα γραφικά στοιχεία, που βρίσκονται σε κοινή χρήση σε όλη την ιστοσελίδα, όπως για παράδειγμα το φόντο. Η παρουσίαση μπορεί να σχεδιαστεί από ένα πρόγραμμα γραφικού σχεδιασμού, το οποίο μπορεί να είναι ενσωματωμένο στο CMS, ή να χρειάζεται χειροκίνητο προγραμματισμό σε γλώσσες, όπως η HTML, CSS και άλλες γλώσσες για την δημιουργία script. Σε κάθε περίπτωση, δημιουργούνται από τους προγραμματιστές του CMS πρότυπα ή «συσκευασίες» παρουσίασης(packages), οι οποίες μπορούν να εφαρμοστούν σε όλη ή σε μέρος της ιστοσελίδας.

Δημιουργία Περιεχομένου: Εργαλεία φορμών και προγραμματισμού HTML σε μορφή WYSIWYG (Αυτό Που Βλέπεις Είναι Αυτό Που Παίρνεις) βοηθούν τους χρήστες του CMS να

προσθέσουν υλικό χωρίς να χρειάζονται να χρησιμοποιήσουν τεχνικούς πόρους. Τα WYSIWYG εργαλεία βοηθούν τους συντάκτες περιεχομένου όχι μόνο να προσθέσουν περιεχόμενο στην ιστοσελίδα χωρίς να χρειάζονται πολλές τεχνικές γνώσεις, αλλά και να παράγουν ένα άρτιας σχεδίασης τελικό προϊόν. Ένα CMS συνήθως περιλαμβάνει μία μεγάλη ποικιλία λειτουργιών, όπως για παράδειγμα: την εισαγωγή γραφικών, μορφοποίηση κειμένου (γραμματοσειρά, μέγεθος, χρώμα, υπογράμμιση, πλάγια κ.λ.π.), δημιουργία πινάκων, ορθογραφικό έλεγχο κ.α. Η λειτουργία προεπισκόπησης επιτρέπει φυσικά στον χρήστη να ελέγξει ξανά το περιεχόμενο μέσα στα πλαίσια της παρουσίασης του, πριν αυτό δημοσιευτεί στην ιστοσελίδα. Αυτό είναι ένα από τα βασικά πλεονεκτήματα των CMS.

Εργαλεία Ροής: πρόκειται για εργαλεία, που βοηθούν την αυτοματοποίηση της διαδικασίας της ροής του περιεχομένου κατά την διάρκεια της παραγωγής του. Σε ένα CMS εμφανίζονται συνήθως τρία στάδια, do ρόλος του σχεδιαστή, ο ρόλος του συντάκτη/διορθωτή και ο ρόλος του εκδότη, που δίνει την έγκριση του. Μία νέα ή μία αναμορφοποιημένη σελίδα περνάει από κάθε στάδιο, πολλές φορές περισσότερες από μία φορές, μέχρι να δημοσιευτεί. Μικρότεροι ή λιγότερο σύνθετοι οργανισμοί χρησιμοποιούν συχνά μία απλούστερη προσέγγιση ενός σταδίου, για τη δημοσίευση του περιεχομένου. Ο κάθε συντάκτης, δηλαδή, δημοσιεύει ο ίδιος το περιεχόμενο του απευθείας στην ιστοσελίδα, ενσωματώνοντας στο πρόσωπο του και τους τρεις ρόλους.

Εικόνα 11. Τυπικό διάγραμμα ροής εργασίας σε ένα CMS [17]

Φόρμες Βάσεων Δεδομένων: πρόκειται για φόρμες, που εμφανίζονται στην δημοσιευμένη ιστοσελίδα και χρησιμοποιούνται για την επί τόπου συγκέντρωση στοιχείων από τους επισκέπτες της. Αυτές οι φόρμες χειρίζονται τις βασικές ανάγκες στην συλλογή δεδομένων, όπως μία σελίδα για την εγγραφή εθελοντών. Τα περισσότερα CMS προσφέρουν την δυνατότητα σε χρήστες χωρίς πολλές τεχνικές γνώσεις να ρυθμίσουν αυτές τις φόρμες.

Εργαλεία Αναζήτησης: πρόκειται για εργαλεία, που επιτρέπουν την αναζήτηση χαρακτηριστικών στοιχείων τόσο σε όλο το μήκος τον δικτυακό τόπο, όσο και σε κάποια συγκεκριμένη περιοχή, που καθορίζεται από τον χρήστη. Επίσης, αφορά τα εργαλεία αναζήτησης κειμένου από τους επισκέπτες της ιστοσελίδας, τα οποία την κάνουν πιο προσβάσιμη και εύχρηστη. Τα καλύτερα εργαλεία αναζήτησης ερευνούν στα κείμενα και στις σελίδες του δικτυακού τόπου και προσφέρουν λειτουργίες σύνθετης αναζήτησης. Τέλος, πρέπει να σημειωθεί ότι, για να βελτιώσουν τα αποτελέσματα της αναζήτησης, οι διαχειριστές χρησιμοποιούν συχνά ειδικά εργαλεία, τα οποία τεμαχίζουν ή κατηγοριοποιούν τα κείμενα, τα αρχεία και τις εικόνες, διευκολύνοντας με αυτό τον τρόπο την αναζήτηση τους.

Εργαλεία Ενσωμάτωσης: πρόκειται για πολύ μικρές εφαρμογές, που υποστηρίζουν την γρήγορη διασύνδεση ανάμεσα στα CMS και στα συστήματα διαχείρισης οικονομικών δεδομένων, όπως είναι, για παράδειγμα, της λογιστικής, της διαχείρισης μελών και δωρεών, των τραπεζικών συναλλαγών και του ηλεκτρονικού εμπορίου.

Αναφορές Ιστοσελίδας: πρόκειται για στατιστικά στοιχεία που συγκεντρώνονται από ένα CMS, ώστε ο διαχειριστής να έχει καλύτερη επίβλεψη. Οι αναφορές αυτές αναλύουν την καθημερινή κίνηση της ιστοσελίδας, τις σελίδες που συνάντησαν το μεγαλύτερο ενδιαφέρον από τους επισκέπτες, την προέλευση των επισκεπτών, την μέση διάρκεια των επισκέψεων στην ιστοσελίδα. Ακόμη, αναφέρουν τον πιο συχνό όρο που αναζητήθηκε από τα εργαλεία αναζήτησης, πια μέθοδος αναζήτησης χρησιμοποιήθηκε περισσότερο, αλλά και άλλα στατιστικά στοιχεία.

Σχεδόν κάθε εργαλείο από τα προηγούμενα είναι διαθέσιμο στα περισσότερα open source CMS. Παρόλα αυτά, η προηγούμενη συλλογή εργαλείων αποτελεί την αρχική σύνθεση ενός CMS, που απευθύνεται σε αρχάριους στον χώρο ή σε οργανισμούς, που επιζητούν μία απλά αξιοπρεπή παρουσία στον χώρο του Διαδικτύου. Περισσότερο εξελιγμένες και σύνθετες λύσεις, καθώς επίσης και χαρακτηριστικά, τα οποία δεν είναι απαραίτητα σε κάθε ιστοσελίδα, παρέχονται από εξειδικευμένα CMS. Τα χαρακτηριστικά αυτά βρίσκονται ενσωματωμένα στο CMS, δηλαδή για τον χειρισμό τους είναι υπεύθυνος και πάλι ο διαχειριστής του προγράμματος, μοιράζονται τον ίδιο πίνακα ελέγχου και έχουν κοινή βάση δεδομένων με τα βασικά χαρακτηριστικά, στα CMS στα οποία προσφέρονται.

3.6 Είδη Συστημάτων Διαχείρισης Περιεχομένου

Τα Content Management Systems διακρίνονται σε ορισμένες κατηγορίες ανάλογα με ορισμένα βασικά χαρακτηριστικά τα οποία παρουσιάζουν. Μπορούν, λοιπόν, να κατηγοριοποιηθούν ανάλογα με το είδος του παρόχου τους και ανάλογα με το που βρίσκεται ο χώρος αποθήκευσης και διαχείρισης της βάσης δεδομένων και του CMS [17].

3.6.1 ASP και Licensed (με βάση το χώρο αποθήκευσης και διαχείρισης)

Στα **Application Service Provider (ASP)** CMS, δηλαδή Υποστήριξης Παρόχου Υπηρεσίας, ο κατασκευαστής τους φιλοξενεί όλα τα δεδομένα και το λογισμικό στους server της εταιρίας του. Με αυτόν τον τρόπο απαλείφονται τα έξοδα για μία ακριβή αγορά λογισμικού και hardware του συστήματος, που θα φιλοξενεί το CMS. Παράλληλα μειώνονται και οι ανάγκες για τεχνικούς πόρους, όπως για παράδειγμα για συντηρητές του δικτύου των υπολογιστών. Τέλος, βασικότερο πλεονέκτημα ενός τέτοιου είδους συστήματος είναι η συνεχής εξέλιξη, καθώς ο πάροχος προωθεί διαρκώς νέες λειτουργίες του προϊόντος και ανανεώσεις στου πελάτες του, προσφέροντας έτσι το χαρακτηριστικό της άμεσης ανανέωσης και πρωτοπορίας της ιστοσελίδας.

Στα CMS με **παροχή άδειας (Licensed),** ο πάροχος του πουλάει το προϊόν, δηλαδή παρέχει άδεια χρήσης του, δεν εμπλέκεται στην όλη διαδικασία λειτουργίας του και ο χρήστης είναι πλέον υπεύθυνος, ώστε να το εγκαταστήσει, να το ρυθμίσει και να το συντηρήσει. Διαχειριστής σε αυτήν την περίπτωση είναι το τεχνικό τμήμα του οργανισμού. Η προσέγγιση αυτών των CMS εξασφαλίζει ότι φιλοξενείς και διαχειρίζεσαι τα δικά σου δεδομένα. Επίσης, τα Licensed είναι ιδανικά για οργανισμούς, οι οποίοι διατηρούν ήδη στις εγκαταστάσεις του κάποιο είδος υπηρεσίας παρόμοιας, όπως για παράδειγμα το σύστημα Διαχείρισης Εξυπηρέτησης Πελατών (CRM), οπότε θα ήταν πιο φθηνό να συντηρούν ταυτόχρονα και ένα CMS.

3.6.2 Commercial, Open source, Managed Open Source (με βάση το είδος του παρόχου)

Commercial: πρόκειται για λογισμικό, που προέρχεται είτε από κερδοσκοπικές είτε από μη κερδοσκοπικές εταιρίες. Οι πάροχοι αυτοί αναπτύσσουν κατά κύριο λόγο το λογισμικό, το οποίο στην συνέχεια πουλάνε και υποστηρίζουν τεχνικά. Στην σημερινή εποχή, οι εμπορικές αυτές λύσεις είναι πιο συχνές από τις ελεύθερες λύσεις των open source CMS.

Open Source: πρόκειται για μία λύση CMS, που δημιουργείται και συντηρείται από έναν ανεπίσημο και ανιδιοτελή συνεργάτη μίας κοινότητας χρηστών. Στην συνέχεια, το λογισμικό αυτό διανέμεται για συγκεκριμένο σκοπό στα μέλη αυτής της κοινότητας. Για αυτά τα ανοιχτά λογισμικά θα πρέπει σαφώς στο κόστος τους να συμπεριληφθεί και τα έξοδα τεχνικής υποστήριξης τους, τα οποία σαφώς και είναι αυξημένα σε αυτό το μοντέλο. Ακόμη, θα πρέπει να προστεθεί το εσωτερικό hardware και λογισμικό και το τεχνικό προσωπικό που χρειάζεται για να συντηρηθεί αυτό το σύστημα, όπως είναι για παράδειγμα οι προγραμματιστές, οι οποίοι εγκαθιστούν τις ανανεώσεις και εξελίσσουν τις λειτουργίες του προγράμματος.

Managed Open Source: πρόκειται για έναν συνδυασμό της εμπορικής και της ελεύθερης προσέγγισης, όπου ένας πάροχος υιοθετεί μία open- source λύση σαν την βασική του πλατφόρμα και στην συνέχεια προσφέρει την λύση αυτή σε άλλους σε συνδυασμό με συμπληρωματικές υπηρεσίες τεχνικής υποστήριξης. Αυτή η λύση ουσιαστικά σχεδόν δεν υπάρχει σήμερα στην κοινότητα των μη-κερδοσκοπικών παρόχων. Παρόλα αυτά, καθώς οι λύσεις open-source ωριμάζουν, οι ειδικοί περιμένουν ότι θα εμφανιστούν πολύ πιο έντονα.Όσον αφορά τη διάκριση των CMS σε σχέση με τον τρόπο παράδοσης τους, έχουμε δύο μορφές λογισμικού. Υπάρχουν εκατοντάδες επιλογές από CMS και των δύο κατηγοριών και η κάθε μία από αυτές διαφέρει στην υλοποίηση, στο κόστος και στην εξυπηρέτηση.

3.7 Κριτήρια Επιλογής CMS

Ένα CMS αποτελεί για τους περισσότερους οργανισμούς, οποιουδήποτε μεγέθους, μία αγορά κεφαλαίου. Επειδή, λοιπόν, οι λύσεις που προσφέρονται στην διαχείριση περιεχομένου είναι πολλαπλές και πολλές φορές πολύπλοκες και εξειδικευμένες, υπάρχουν κάποιοι παράγοντες που πρέπει να λάβει κανείς υπόψη πριν αγοράσει, κατεβάσει από το Διαδίκτυο και εγκαταστήσει ένα CMS. Τα κριτήρια επιλογής του λογισμικού αυτού, επομένως θα πρέπει να είναι τα εξής [17]:

Open Source ή Commercial: στην επιλογή αυτή σημαντικό παράγοντα παίζει η έννοια κόστος. Στην περίπτωση του open-source λογισμικού, αυτό παρέχεται «δωρεάν». Στην πραγματικότητα, όμως, κρύβει κόστη σχετικά με την τεχνική υποστήριξη του. Τα ερωτήματα, που πρέπει να απαντηθούν είναι, ποιος θα υποστηρίζει τεχνικά το λογισμικό και ποιος θα δημιουργεί νέες λειτουργίες και θα εγκαθιστά τις ανανεώσεις. Χρειάζεται, άρα, μεγάλη προσοχή, καθώς υπάρχουν πολλές βιώσιμες open-source λύσεις, αλλά καλό θα ήταν πάντα να συνυπολογίζεται το συνολικό κόστος.

ASP ή Licensed: στην επιλογή αυτή σημαντικό παράγοντα παίζει το που θα εγκατασταθεί το λογισμικό και η βάση δεδομένων. Υπάρχουν οργανισμοί, που προτιμούν να έχουν τον άμεσο έλεγχο της ιστοσελίδας και των δεδομένων τους, και να φιλοξενούν για το λόγο αυτό το CMS στις εγκαταστάσεις τους. Άλλοι οργανισμοί, για να γλιτώσουν το διαχειριστικό κόστος, αναθέτουν την εγκατάσταση και την συντήρηση του CMS σε εξωτερικούς συνεργάτες. Τα ερωτήματα, που τίθενται, είναι: υπάρχει το

απαραίτητο προσωπικό, για να αντιμετωπίσει τα προβλήματα που μπορεί να προκύψουν τις πλέον ακατάλληλες ώρες, όπως πολύ αργά το βράδυ; Υπάρχει ο εξοπλισμός, που χρειάζεται για την συντήρηση του λογισμικού, όπως για παράδειγμα στην περίπτωση φιλοξενίας και κάποιου άλλου συστήματος, με αντίστοιχες προϋποθέσεις και κόστος; Χρειάζεται, τέλος και σε αυτή την περίπτωση να λαμβάνεται υπόψη το γεγονός, ότι η διαχείριση του περιεχομένου στο Διαδίκτυο είναι από τις πιο χρονικά ευαίσθητες λειτουργίες, λόγω της φύσης του μέσου, που προσφέρεται για γρήγορη παροχή πληροφοριών και περιεχομένου.

Χρήστες - Συντάκτες Περιεχομένου: στο κριτήριο αυτό απαιτείται να υπολογιστεί ρεαλιστικά ο αριθμός των χρηστών, που θα συνεισφέρουν στην ιστοσελίδα. Παράλληλα, σημαντικό ρόλο παίζει και το είδος του περιεχομένου, για το οποίο ο καθένας από αυτούς θα είναι υπεύθυνος, και αν το περιεχόμενο αυτό θα δημοσιεύεται άμεσα στον δικτυακό τόπο ή θα περνάει από τον έλεγχο κάποιου αρχισυντάκτη. Οι επιλογές αυτές θα βοηθήσουν στην επιλογή CMS, τα οποία θα προσφέρουν δυναμική και ασφαλή λειτουργία ροής περιεχομένου και δυνατότητα διαχείρισης και ελέγχου ενός μεγάλου αριθμού συντακτών, αν αυτό χρειάζεται.

Είδη Περιεχομένου: το κριτήριο αυτό αφορά τις μορφές του υλικού, που θα εμφανίζεται στην ιστοσελίδα. Οι περισσότερες εφαρμογές διαχειρίζονται κείμενα, γραφικά και φωτογραφίες. Αν, όμως, στο δικτυακό τόπο δημοσιεύεται υλικό με πλούσια μορφοποίηση, όπως για παράδειγμα με πλάγια, έντονα, υπογραμμισμένα και μαρκαρισμένα γράμματα, ή πίνακες και λίστες με κουκίδες, συλλογές φωτογραφιών και ήχος ή βίντεο streaming, τότε, το CMS, που θα επιλεγεί, θα πρέπει να προσφέρει αυτές τις δυνατότητες.

Μονάδα Συσχετιζόμενων Συστημάτων: το κριτήριο αυτό αφορά την ύπαρξη κάποιας μορφής διασύνδεσης ανάμεσα στο CMS και στα υπόλοιπα συστήματα, όπως αυτό των χορηγιών από τους επισκέπτες της ιστοσελίδας (Donation System), του συστήματος εξυπηρέτησης πελατών και του συστήματος των ηλεκτρονικών πωλήσεων. Όλες αυτές οι διασυνδέσεις είναι διαθέσιμες από κάποιους εξειδικευμένους παρόχους, οι οποίοι συνεργάζονται με εταιρίες που σχεδιάζουν τα παραπάνω συστημάτων, ώστε να υπάρχει η κατάλληλη συνεργασία. Παράλληλα, σημαντικός παράγοντας είναι και η εξέλιξη ενός δικτυακού τόπου, αφού καθώς αυτή ωριμάζει, οι σχέσεις μεταξύ των εμπλεκόμενων συστημάτων και βάσεων δεδομένων γίνεται σαφώς πιο πολύπλοκη.

Αναφορές: το κριτήριο αυτό αφορά το είδος των στατιστικών αποτελεσμάτων, που θα αναφέρει το CMS. Σε περίπτωση που απαιτούνται ιδιαίτερες αναφορές, όπως αυτές που χρειάζονται από τους υπεύθυνους των μελών, των χορηγών και της επικοινωνίας, τότε θα πρέπει να υποστηρίζονται από το επιλεγμένο CMS, ώστε να αξίζει η επένδυση σ' αυτό.

Επανασχεδίαση ή Μετακίνηση: το κριτήριο αυτό αφορά την πιθανότητα ανασχεδίασης του δικτυακού χώρου και την μετακίνηση στοιχείων, που θα χρησιμοποιηθούν από την παλιά ιστοσελίδα. Η δυνατότητα εύκολης μετακίνησης του κώδικα και των δεδομένων της ιστοσελίδας είναι πολύ σημαντική σε αυτήν την περίπτωση.

Πολυπλοκότητα Εμφάνισης: το κριτήριο αυτό αφορά την υποστήριξη από το λογισμικό της πολύπλοκης παρουσίασης του δικτυακού τόπου. Όταν η ιστοσελίδα περιέχει δυναμικά μενού πλοήγησης, στοιχεία Flash, ή άλλες σύνθετες γλώσσες γραφικού σχεδιασμού, χρειάζεται ένα πιο σύνθετο σύστημα διαχείρισης περιεχομένου.

3.8 Επίλογος

Για την υλοποίηση της παρούσας εργασίας, υιοθετήθηκε η λογική των CMS η οποία παρουσιάστηκε στο κεφάλαιο αυτό. Στο επόμενο κεφάλαιο παρουσιάζονται αναλυτικά οι τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη του CMS.

Κεφάλαιο 4ο: Τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη του CMS με MongoDB

4.1 Εισαγωγή

Στο κεφάλαιο αυτό παρουσιάζονται οι τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη του CMS με χρήση MongoDB, Η Expressjs η οποία είναι ένα web framework της Nodejs και η MongoDB ως βάση δεδομένων.

4.2 Node.js

Η Node.js [18] είναι ένα ανοιχτού κώδικα, cross-platform runtime περιβάλλον το οποίο δίνει τη δυνατότητα στους προγραμματιστές να δημιουργούν όλων των ειδών server-side εργαλεία και εφαρμογές σε Javascript. Το runtime περιβάλλον παραλείπει οποιοδήποτε Javascript API για browser και υποστηρίζει περισσότερο τα παραδοσιακά OS APIs, όπως το HTTP και οι βιβλιοθήκες συστήματος αρχείων (file system libraries).

Από την πλευρά της ανάπτυξης web server, η Node έχει πολλαπλά πλεονεκτήματα όπως:

Υψηλή Αποδοτικότητα: Η Node έχει σχεδιαστεί ώστε να βελτιστοποιεί την απόδοση και την επεκτασιμότητα σε web εφαρμογές και είναι μία καλή λύση σε πολλά κοινά προβλήματα του web development, όπως για παράδειγμα η ανάπτυξη web εφαρμογών πραγματικού χρόνου.

Ο κώδικας γράφεται σε απλή Javascript και αυτό σημαίνει ότι απαιτείται λιγότερος χρόνος όταν αναπτύσσεται και η πλευρά του πελάτη και η πλευρά του εξυπηρετητή (client side & server side) και απαιτείται η εναλλαγή από τη μία γλώσσα προγραμματισμού σε άλλη.

Η Javascript είναι μία σχετικά καινούργια γλώσσα προγραμματισμού και η σχεδίασή της είναι πιο βελτιστοποιημένη σε σχέση με άλλες παραδοσιακές γλώσσες για web-server (πχ Python, PHP κλπ). Υπάρχουν επίσης πολλές άλλες γλώσσες οι οποίες γίνονται compiled και μετατρέπονται σε Javascript, όπως για παράδειγμα η TypeScript, η CoffeeScript, η Scala κλπ

Το **NPM** (node package manager) παρέχει πρόσβαση σε πολλές εκατοντάδες επαναχρησιμοποιήσιμα πακέτα.

Η Nodejs είναι **portable** και είναι διαθέσιμη στα περισσότερα λειτουργικά συστήματα όπως Microsoft Windows, macOS, Linux, Solaris κλπ. Υποστηρίζεται επίσης από πολλούς παρόχους web hosting οι οποία συχνά παρέχουν ειδική υποδομή και documentation για τη φιλοξενία Node ιστοσελίδων.

Έχει πολύ ενεργό **third-party** οικοσύστημα και μεγάλη κοινότητα προγραμματιστών, με πολλούς ανθρώπους οι οποίο είναι διαθέσιμη για να βοηθήσουν ο ένας τον άλλον.

4.3 Express

Η Express [18] είναι το πιο διαδεδομένο Node web framework και αποτελεί την κύρια βιβλιοθήκη και για άλλα Node web frameworks. Παρέχει μηχανισμούς όπως:

- Write handlers για διάφορες HTTP μεθόδους σε διαφορετικά URL μονοπάτια. Τα μονοπάτια αυτά στην Express ονομάζονται routes.
- Ενσωματώνει μηχανισμούς για view rendering με σκοπό την παραγωγή responses εισάγοντας δεδομένα μέσα σε templates.
- Ορίζει βασικές ρυθμίσεις για ένα web application, όπως η πόρτα (port) που χρησιμοποιείται για τη σύνδεση και την τοποθεσία των προτύπων που χρησιμοποιούνται για το rendering ενός response.
- Προσθέτει επιπλέον middlewares που διαχειρίζονται τα requests

Σε οποιαδήποτε data-driven ιστοσελίδα, ένα web application περιμένει για http requests από τον web browser ή άλλους clients. Όταν λαμβάνεται το request, η εφαρμογή εκτελεί τις απαραίτητες ενέργειες ανάλογα με το URL pattern και πιθανή πληροφορία που περιέχεται στα POST ή GET δεδομένα. Ανάλογα με τις εκάστοτε απαιτήσεις μπορεί έπειτα να χρειαστεί να γράψει ή να διαβάσει πληροφορίες από μία βάση δεδομένων. Η εφαρμογή στη συνέχεια επιστρέφει ένα response στον web browser και δημιουργεί δυναμικά μία HTTP σελίδα εισάγοντας τα δεδομένα σε placeholders μέσα σε ένα HTTP πρότυπο.

Η Express παρέχει μεθόδους που προσδιορίζουν ποια HTTP μέθοδος καλείται (GET, POST, SET κλπ), το URL pattern (route), μεθόδους οι οποίες προσδιορίζουν ποιος μηχανισμός προτυποποίησης (view) χρησιμοποιείται, πού έχουν τοποθετηθεί τα αρχεία των προτύπων και τι πρότυπο θα χρειαστεί για να γίνει το render ενός response.

Μπορούν επιπρόσθετα να χρησιμοποιηθούν middlewares της Express για την υποστήριξη cookies, χρηστών και sessions κλπ και οποιοσδήποτε μηχανισμός βάσης δεδομένων υποστηρίζεται από τη Node. Σε αυτό το σημείο είναι σημαντικό να αναφέρουμε ότι η Express δεν ορίζει συμπεριφορές που σχετίζονται με βάσεις δεδομένων. [18]

Κεφάλαιο 5ο: Βασικές Λειτουργίες του CMS που αναπτύχθηκε και πώς υλοποιήθηκαν

5.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα δείξουμε πώς υλοποιήθηκε η κάθε λειτουργία στην εφαρμογή.

5.2 Ρόλοι Χρηστών

Στην εφαρμογή που αναπτύχθηκε, υπάρχουν τρεις ρόλοι χρηστών:

Ρόλος Επισκέπτη: Επισκέπτης της εφαρμογής θεωρείται οποιοσδήποτε χρησιμοποιεί την εφαρμογή χωρίς να έχει κάνει εγγραφή και είσοδο με τα στοιχεία σύνδεσής του. Οι επισκέπτες έχουν τη δυνατότητα να διαβάσουν τα άρθρα που έχουν δημιουργηθεί και να αφήσουν σχόλια σε αυτά.

Ρόλος Συγγραφέα: Συγγραφείς της εφαρμογής θεωρούνται όσοι χρήστες έχουν κάνει εγγραφή και έπειτα σύνδεση. Οι συγγραφείς έχουν το δικό τους διαχειριστικό κομμάτι της εφαρμογής και δικό τους dashboard. Έχουν τη δυνατότητα να δημιουργούν νέα άρθρα και να επεξεργάζονται ήδη υπάρχοντα. Μπορούν επίσης να δημιουργήσουν κατηγορίες και να κατηγοριοποιήσουν αντίστοιχα το κάθε άρθρο.

Ρόλος Διαχειριστή: Οι διαχειριστές της εφαρμογής έχουν δικό τους dashboard, διαφορετικό από αυτό των συγγραφέων, μέσω του οποίο μπορούν να εγκρίνουν ή όχι τα σχόλια των επισκεπτών, να διαγράψουν άρθρα ή κατηγορίες και να δημιουργήσουν/επεξεργαστούν άρθρα.

5.3 Δομή του Project

Ο φάκελος του Project, οι υποφάκελοι και τα αρχεία, διαμορφώθηκαν ως εξής:

My CMS

---**config**

------configuration.js

------customFunctions.js

---**controllers**

------adminController.js

------authorController.gr

------defaultController.js

---**models**

------CategoryModel.js

------CommentModel.js

------PostModel.js

------UserModel.js

---**node_modules**

---**public**

------assets

---------BootStrap templates.zip

------css

---------blog-home.css

---------blog-post.css

---------bootstrap-switch.css

---------sb-admin.css

------js

---------bootstap-switch.js

---------sb-admin.js

---------sb-admin-charts.js

---------sb-admin-datatables.js

------uploads

------vendor

---------bootstrap

---------chart.js

---------datatables

---------font-awesome

---------jquery

---------jquery-easing

---**routes**

------adminRoutes.js

------authorRoutes.js

------defaultRoutes.js

---**views**

------admin

------author

------default

------layouts

------partials

---app.js

---package.json

Φάκελος **config**: Περιέχει μεθόδους για τη σύνδεση με τη βάση δεδομένων, ορισμό της πόρτας στην οποία «ακούει» η εφαρμογή μας (configuration.js) και ορισμένες custom μεθόδους που χρησιμοποιούνται στο υπόλοιπο project (customFunctions.js)

Φάκελος **public**: Περιέχει τα αρχεία που είναι προσβάσιμα από το κοινό

Φάκελος **views**: Περιέχει τα handlebars για τον κάθε ρόλο ξεχωριστά.

Αρχείο **app.js**: Περιέχει τη σύνδεση στη βάση δεδομένων, την προσθήκη βιβλιοθηκών που χρησιμεύουν στην υλοποίηση του project, ορίζει τις σταθερές (const) για τα routes και εκκινεί τον server.

Αρχείο **package.json**: Δημιουργείται κατά την εκτέλεση του npm init και περιέχει τις βασικές ιδιότητες του project, όπως name, versions, author, license, ορισμός main κλάσης (στην περίπτωση αυτή app.js)

5.4 Σχήμα Βάσης Δεδομένων

Το σχήμα της MongoDB βάσης δεδομένων που δημιουργήθηκε και χρησιμοποιήθηκε για την παρούσα εργασία αποτελείται από τέσσερις συλλογές (collections):

- categories
- comments
- posts
- users

Τα έγγραφα της συλλογής **categories** έχουν την παρακάτω δομή:

{

“**_id**”: ObjectId(“5f5e5ec3a75f451494590a3b”)

“**title**”: “Tech”

}

Τα έγγραφα της συλλογής **comments** έχουν την παρακάτω δομή:

{

“**_id**”: ObjectId(“5f5e5efda75f451494590a3c”)

“**date**”: 2020-09-13T17:42:21.479+00:00

“**commentIsApproved**”: true

“**body**”: “Test Comment”

}

Τα έγγραφα της συλλογής **posts** έχουν την παρακάτω δομή:

{

"**_id**": ObjectId("5f5e5a55a75f451494590a39")

"**status**": "public"

"**creationDate**": 2020-09-13T17:42:21.405+00:00

"**comments**": [

0: ObjectId("5f5e5efda75f451494590a3c")

1: ObjectId("5f5e5f05a75f451494590a3e")

]

"**allowComments**": false

"**file**": "/uploads"

"**title**": "Post 1"

"**description**": "Post 1 Content"

}

Τα έγγραφα της συλλογής **users** έχουν την παρακάτω δομή:

{

"**_id**": ObjectId("5f5e5a35a75f451494590a38")

"**firstName**": "Eva"

"**lastName**": "Pal"

"**email**": "epalaioc@gmail.com"

"**password**": "$2a$10$sbaBXZBRq0lxY3HPhI7KouIfuKFLyg.5fnjzM7i38.LUf5.Ipcj6q"

}

5.5 Σύνδεση στη βάση δεδομένων

Για τη σύνδεση με τη MongoDB, χρησιμοποιείται η connect() μέθοδος της Mongoose, η οποία λαμβάνει ως παράμετρο το URL της MongoDB, προσπαθεί να συνδεθεί στη βάση δεδομένων και τυπώνει τα αντίστοιχα μηνύματα.

```
// Configure Mongoose to Connect to MongoDB
mongoose.connect(mongoDbUrl, { useNewUrlParser: true })
    .then(response => {
        console.log("MongoDB Connected Successfully.");
    }).catch(err => {
        console.log("Database connection failed.");
});
```

Το URL της MongoDB, έχει οριστεί στο αρχείο /config/configuration.js ως εξής:

```
module.exports = {
    mongoDbUrl : 'mongodb://localhost:27017/my_cms',
    PORT: process.env.PORT || 3000,
```

5.6 Εκκίνηση του web server

Για την εκκίνηση του web server, χρησιμοποιείται η μέθοδος listen() η οποία λαμβάνει ως παράμετρο το PORT, δηλαδή την πόρτα στην οποία θα «ακούει» η εφαρμογή. Το PORT, έχει οριστεί στο αρχείο /config/configuration.js όπως φαίνεται και στην παραπάνω ενότητα.

```
app.listen(PORT, () => {
    console.log(`Server is running on port ${PORT}`);
});
```

5.7 Index page

Κατά την εκκίνηση της εφαρμογής, ο επισκέπτης μπορεί να δει την κεντρική σελίδα μέσω του defaultController και της μεθόδου index η οποία κάνει render τη σελίδα που βρίσκεται στο directory /views/default/index μαζί με όλα τα posts και τις κατηγορίες.

```
router.all('/*', (req, res, next) => {

    req.app.locals.layout = 'default';

    next();
});


router.route('/')
    .get(defaultController.index);
```

```
index: async (req, res) => {

    const posts = await Post.find();
    const categories = await Category.find();
    res.render('default/index', {posts: posts, categories: categories});
},
```

5.8 Read More

Όταν ένας επισκέπτης επιλέξει Read More σε ένα άρθρο, καλείται η παρακάτω μέθοδος η οποία δέχεται ως παράμετρο το id του post που υπάρχει στο request που έγινε και με τη βοήθεια της μεθόδου findById(id) προσπαθεί να βρει το συγκεκριμένο post μέσα στη βάση δεδομένων.

Στη συνέχεια συμπληρώνεται το όνομα του χρήστη που το δημιούργησε το κείμενο του άρθρου και τα σχόλια που έχουν γίνει και έχουν εγκριθεί από τον διαχειριστή.

```
getSinglePost: (req, res) => {
    const id = req.params.id;

    Post.findById(id)
        .populate({path: 'comments', populate: {path: 'user', model:
'user'}})
        .then(post => {
        if (!post) {
            res.status(404).json({message: 'No Post Found'});
        }
        else {
            res.render('default/singlePost', {post: post, comments:
post.comments});
```

```
        }
    })
},
```

5.9 Comment

Όταν ο επισκέπτης πληκτρολογήσει ένα σχόλιο σε ένα συγκεκριμένο post και επιλέξει Submit, τότε η μέθοδος submitComment δέχεται το id του comment που έγινε και δημιουργεί ένα νέο comment στη βάση δεδομένων συσχετίζοντάς το με το αντίστοιχο άρθρο.

```
submitComment: (req, res) => {

        Post.findById(req.body.id).then(post => {
            const newComment = new Comment({
                body: req.body.comment_body
            });

            post.comments.push(newComment);
            post.save().then(savedPost => {
                newComment.save().then(savedComment => {
                  req.flash('success-message', 'Your comment was submitted
for review.');
                  res.redirect(`/post/${post._id}`);
                });
            });

        });


}
```

5.10 Approve Comments

Από την πλευρά του, ο διαχειριστής αφού εισέλθει στο dashboard του και επιλέξει Comments, μπορεί να δει όλα τα σχόλια, είτε αυτά έχουν γίνει ήδη approved είτε όχι.

```
getComments: (req, res) => {
    Comment.find()
        .populate('user')
        .then(comments => {
            res.render('admin/comments/index', {comments: comments});
        })
},
```

Όταν τα σχόλια γίνονται submit, είναι από προεπιλογή not approved. Όταν ο διαχειριστής επιλέξει να κάνει approve ένα σχόλιο, τότε καλείται η παρακάτω μέθοδος η οποία δέχεται σαν παράμετρο το id του σχολίου και μέσω της commentIsApproved το σχόλιο εγκρίνεται.

```
approveComments: (req, res) => {
    var data = req.body.data;
    var commentId = req.body.id;

    console.log(data, commentId);

    Comment.findById(commentId).then(comment => {
        comment.commentIsApproved = data;
        comment.save().then(saved => {
            res.status(200).send('OK');
```

```
        }).catch(err => {
            res.status(201).send('FAIL');
        });
    });
}
```

```
commentIsApproved: {
    type: Boolean,
    default: false
}
```

5.11 Register

Για την υλοποίηση του Register χρησιμοποιήθηκε η παρακάτω μέθοδος η οποία αρχικά μόλις γίνει click στο κουμπί Register κάνει render τη φόρμα για το registration, η οποία υπάρχει στο directory views/default/register με τη μορφή handlebars.

```
registerGet: (req, res) => {
    res.render('default/register');
},
```

Μόλις ο χρήστης κάνει κλικ στο κουμπί Register, χρησιμοποιείται η παρακάτω μέθοδος η οποία αρχικά «κοιτάζει» εάν κάποιο από τα βασικά πεδία (First name, Last name, Email, Password) δεν έχει συμπληρωθεί και «ρίχνει» το αντίστοιχο σφάλμα/σφάλματα κάνοντάς τα render στη ίδια σελίδα (views/default/register).

Εάν δεν υπάρξουν σφάλματα, δηλαδή έχουν συμπληρωθεί όλα τα πεδία, μέσω της μεθόδου findOne() της MongoDB ελέγχεται εάν το συγκεκριμένο email υπάρχει ήδη. Εάν υπάρχει, που σημαίνει ότι ένας author με αυτό το email έχει ήδη δημιουργηθεί, τότε εμφανίζεται το αντίστοιχο σφάλμα και η σελίδα γίνεται redirect στη default/login.

Εάν δεν υπάρχει χρήστης με το ίδιο email τότε ο κωδικός πρόσβασης γίνεται hashed με τη βοήθεια της συνάρτησης bcrypt, δημιουργείται ο νέος χρήστης, εμφανίζεται το μήνυμα “You are now registered” και ο χρήστης προτρέπεται να κάνει login μόλις η σελίδα login γίνει render.

```
registerPost: (req, res) => {
    let errors = [];

    if (!req.body.firstName) {
        errors.push({message: 'First name is mandatory'});
    }
    if (!req.body.lastName) {
        errors.push({message: 'Last name is mandatory'});
    }
    if (!req.body.email) {
        errors.push({message: 'Email field is mandatory'});
    }
    if (!req.body.password || !req.body.passwordConfirm) {
        errors.push({message: 'Password field is mandatory'});
    }
    if (req.body.password !== req.body.passwordConfirm) {
        errors.push({message: 'Passwords do not match'});
    }

    if (errors.length > 0) {
        res.render('default/register', {
```

```
            errors: errors,
            firstName: req.body.firstName,
            lastName: req.body.lastName,
            email: req.body.email
        });
    } else {
        User.findOne({email: req.body.email}).then(user => {
            if (user) {
                req.flash('error-message', 'Email already exists, try to
login.');
                res.redirect('/login');
            } else {
                const newUser = new User(req.body);

                bcrypt.genSalt(10, (err, salt) => {
                    bcrypt.hash(newUser.password, salt, (err, hash) => {
                        newUser.password = hash;
                        newUser.save().then(user => {
                            req.flash('success-message', 'You are now
registered');
                            res.redirect('/login');
                        });
                    });
                });
            }
        });
    }
},
```

5.12 Login

Η λειτουργία του Login υλοποιείται ως εξής:

```
passport.use(new LocalStrategy({
    usernameField: 'email',
    passReqToCallback: true
}, (req, email, password, done) => {
    User.findOne({email: email}).then(user => {
        if (!user) {
            return done(null, false, req.flash('error-message', 'User not
found with this email.'));
        }

        bcrypt.compare(password, user.password, (err, passwordMatched) => {
            if (err) {
                return err;
            }

            if (!passwordMatched) {
                return done(null, false, req.flash('error-message',
'Invalid Username or Password'));
            }

            return done(null, user, req.flash('success-message', 'Login
Successful'));
        });

    });


}));
```

Με τη χρήση της μεθόδου findOne() ελέγχεται αρχικά εάν υπάρχει user με το email που έχει εισάγει ο χρήστης. Εάν υπάρχει, τότε με τη βοήθεια της bcrypt.compare() συγκρίνεται το password που έχει πληκτρολογήσει ο χρήστης, με το password που έχει γίνει hashed κατά το registration. Εάν η είσοδος είναι επιτυχής, τότε ο χρήστης ανακατευθύνεται στο Author Dashboard ως εξής:

```
router.route('/login')
    .get(authorController.loginGet)
    .post(passport.authenticate('local', {
        successRedirect: '/author',
        failureRedirect: '/login',
        failureFlash: true,
        successFlash: true,
        session: true
    }) ,authorController.loginPost);
```

Εάν έχει επιλεχθεί το Admin Login, τότε ο χρήστης ανακατευθύνεται στο Admin Dashboard:

```
router.route('/adminLogin')
    .get(adminController.loginGet)
    .post(passport.authenticate('local', {
        successRedirect: '/admin',
        failureRedirect: '/login',
        failureFlash: true,
        successFlash: true,
        session: true
    }) ,adminController.loginPost);
```

5.13 Create New Post

Όταν ένας author ή administrator επιθυμεί να δημιουργήσει ένα νέο άρθρο, με τη χρήση της παρακάτω μεθόδου γίνεται αρχικά render η σελίδα author/posts/create η οποία περιέχει τη φόρμα για τη δημιουργία νέου άρθρου.

```
getCreatePostPage: (req, res) => {
    Category.find().then(cats => {
        res.render('author/posts/create', {categories: cats});
    });
},
```

Όταν ο χρήστης επιλέξει Create Post, τότε αρχικά ελέγχεται εάν υπάρχει κάποιο κενό πεδίο. Εάν δεν υπάρχει, τότε δημιουργείται ένα νέο άρθρο στη βάση δεδομένων παίρνοντας ως παραμέτρους τις τιμές των πεδίων που όρισε ο χρήστης.

```
submitCreatePostPage: (req, res) => {
    const commentsAllowed = !!req.body.allowComments;
    // Check for any input file
    let filename = '';
    if (!isEmpty(req.files)) {
        let file = req.files.uploadedFile;
        filename = file.name;
        let uploadDir = './public/uploads/';
        file.mv(uploadDir + filename, (err) => {
            if (err)
                throw err;
        });
```

```
    }

    const newPost = new Post({
        title: req.body.title,
        description: req.body.description,
        status: req.body.status,
        allowComments: commentsAllowed,
        category: req.body.category,
        file: `/uploads/${filename}`
    });

    newPost.save().then(post => {
        req.flash('success-message', 'Post created successfully.');
        res.redirect('/author/posts');
    });
},
```

5.14 Edit Post

Όταν ένας author ή administrator επιθυμεί να επεξεργαστεί ένα νέο άρθρο, με τη χρήση της παρακάτω μεθόδου γίνεται αρχικά render η σελίδα author/posts/edit η οποία περιέχει τη φόρμα μαζί με τα συμπληρωμένα πεδία του άρθρου.

```
getEditPostPage: (req, res) => {
    const id = req.params.id;

    Post.findById(id)
        .then(post => {
            Category.find().then(cats => {
                res.render('author/posts/edit', {post: post, categories:
cats});
            });
        });
},
```

Αφού ο χρήστης ολοκληρώσει τις αλλαγές του και επιλέγει Update Post, τότε η παρακάτω μέθοδος λαμβάνει σαν παράμετρο το id του άρθρου, το βρίσκει στη βάση δεδομένων με τη findById() και θέτει τις τιμές των πεδίων του ίσες με τις τιμές που έχει εισάγει ο χρήστης.

```
submitEditPostPage: (req, res) => {
    const commentsAllowed = !!req.body.allowComments;
    const id = req.params.id;
    Post.findById(id)
        .then(post => {
            post.title = req.body.title;
            post.status = req.body.status;
            post.allowComments = commentsAllowed;
            post.description = req.body.description;
            post.category = req.body.category;
            post.save().then(updatePost => {
                req.flash('success-message', `The Post ${updatePost.title}
has been updated.`);
                res.redirect('/author/posts');
            });
        });
},
```

5.15 Delete Post

Όταν ο διαχειριστής επιλέξει Delete σε ένα post μέσω του dashboard του, τότε καλείται η παρακάτω μέθοδος η οποία μέσω της findByIdAndDelete βρίσκει το post με το id του, το διαγράφει και ανακατευθύνει τον διαχειριστεί σε σελίδα posts η οποία εμφανίζει όλα τα άρθρα που έχουν δημοσιευτεί.

```
deletePost: (req, res) => {

    Post.findByIdAndDelete(req.params.id)
        .then(deletedPost => {
            req.flash('success-message', `The post ${deletedPost.title} has
been deleted.`);
            res.redirect('/admin/posts');
        });
},
```

5.16 Categories

Για την προβολή των κατηγοριών, χρησιμοποιείται η μέθοδος find() η οποία βρίσκει όλες τις κατηγορίες που έχουν δημιουργηθεί στη βάση δεδομένων και τις τοποθετεί στη σελίδα που βρίσκεται στο directory views/admin/category/index ή view/author/category/index, ανάλογα με το αν ο συνδεδεμένος χρήστης ή author ή administrator.

```
getCategories: (req, res) => {

    Category.find().then(cats => {
        res.render('admin/category/index', {categories: cats});
    });
},
```

```
getCategories: (req, res) => {

    Category.find().then(cats => {
        res.render('author/category/index', {categories: cats});
    });
},
```

5.17 New Category

Όταν ένας author ή administrator επιλέξει να δημιουργήσει μία νέα κατηγορία, τότε το ακόλουθο κομμάτι κώδικα λαμβάνει τον τίτλο της νέας κατηγορίας, δημιουργεί μία νέα με τον τίτλο αυτό και την εισάγει στη βάση δεδομένων.

```
createCategories: (req, res) => {
    let categoryName = req.body.name;

    if (categoryName) {
        const newCategory = new Category({
            title: categoryName
        });
        newCategory.save().then(category => {
            res.status(200).json(category);
        });
    }

},
```

5.18 Edit Category

Η επεξεργασία μίας κατηγορίας είτε από την πλευρά του συγγραφέα είτε από την πλευρά του διαχειριστή, υλοποιείται όπως φαίνεται στο ακόλουθο κομμάτι κώδικα. Πρακτικά μόλις ο χρήστης επιλέξει Edit, το όνομα της επιλεγμένης κατηγορίας φαίνεται σε ένα text box:

```
getEditCategoriesPage: async (req, res) => {
    const catId = req.params.id;
    const cats = await Category.find();

    Category.findById(catId).then(cat => {
        res.render('admin/category/edit', {category: cat, categories:
cats});
    });
},
```

Μόλις ο χρήστης ολοκληρώσει τις αλλαγές του και επιλέξει Update Category, τότε το παρακάτω κομμάτι κώδικα λαμβάνει το id της συγκεκριμένης κατηγορίας και τον νέο τίτλο. Έπειτα βρίσκει στη βάση δεδομένων την κατηγορία αυτή με τη βοήθεια της μεθόδου findById, θέτει τον τίτλο της ίσο με τον τίτλο που εισήχθησε από τον χρήστη και κάνει render τη σελίδα με όλες τις κατηγορίες..

```
submitEditCategoriesPage: (req, res) => {
    const catId = req.params.id;
    const newTitle = req.body.name;

    if (newTitle) {
        Category.findById(catId).then(category => {
            category.title = newTitle;
            category.save().then(updated => {
                res.status(200).json({url: '/admin/category'});
            });
        });
    }
},
```

5.19 Delete Category

Για τη διαγραφή μίας κατηγορίας, χρησιμοποιείται η παρακάτω μέθοδος, όπου δέχεται σαν παράμετρο το id της κατηγορίας, τη διαγράφει με τη findByIdAndDelete() και έπειτα κάνει render όλες τις κατηγορίες.

```
deleteCategory: (req, res) => {
    Category.findByIdAndDelete(req.params.id)
            .then(deletedCategory => {
            req.flash('success-message', `Category deleted.`);
        });
    Category.find().then(cats => {
        res.render('admin/category/index', {categories: cats});
    });
},
```

5.20 Logout

Όταν ένας author ή ένας admin επιθυμεί να αποσυνδεθεί από την εφαρμογή και επιλέξει Logout, η αποσύνδεσή του υλοποείται ως εξής:

```
router.get('/logout', (req, res) => {
    req.logOut();
    req.flash('success-message', 'Logout was successful');
    res.redirect('/');
});
```

5.21 Επίλογος

Στο κεφάλαιο αυτό παρουσιάστηκε ο τρόπος υλοποίησης της κάθε λειτουργίας με αναφορά στα αντίστοιχα κομμάτια κώδικα του project. Στο επόμενο κεφάλαιο, ορίζονται τα βήματα για την εκτέλεση της εφαρμογής τοπικά σε έναν υπολογιστή.

Κεφάλαιο 6ο: Οδηγίες εκτέλεσης της εφαρμογής τοπικά

6.1 Εισαγωγή

Στο κεφάλαιο αυτό, θα δείξουμε πώς μπορούμε να εκτελέσουμε την εφαρμογή τοπικά σε έναν υπολογιστή. Με τη μέθοδο αυτή, η εφαρμογή θα είναι διαθέσιμη μόνο για τον χρήστη που χρησιμοποιεί τον συγκεκριμένο υπολογιστή. Εάν επιθυμούμε η εφαρμογή να είναι διαθέσιμη στο Internet, τότε θα πρέπει να εφαρμοστούν μέθοδοι port forwarding ή tunneling.

6.2 Εκκίνηση του MongoDB Server

1. Εκτελούμε την γραμμή εντολών των Windows με δικαιώματα διαχειριστή
2. Πληκτρολογούμε την παρακάτω εντολή:
   mongod --port 27017 --dbpath D:\MongoDB\Server\4.2\data\db
   Το path D:\MongoDB\Server\4.2\data\db μπορεί να διαφέρει ανάλογα με το directory στο οποίο έχει αρχικά εγκατασταθεί η MongoDB

6.3 Project Build & Εκκίνηση web server

Χρησιμοποιώντας το Terminal utility του IDE (για την παρούσα εργασία χρησιμοποιήθηκε το WebStorm) πληκτρολογούμε την παρακάτω εντολή npm start.

Στη συνέχεια, ανοίγουμε ένα νέο παράθυρο στον browser μας και στη γραμμή διευθύνσεων πληκτρολογούμε localhost:3000. Στο σημείο αυτό, είμαστε έτοιμοι να χρησιμοποιήσουμε την εφαρμογή.

6.4 Επίλογος

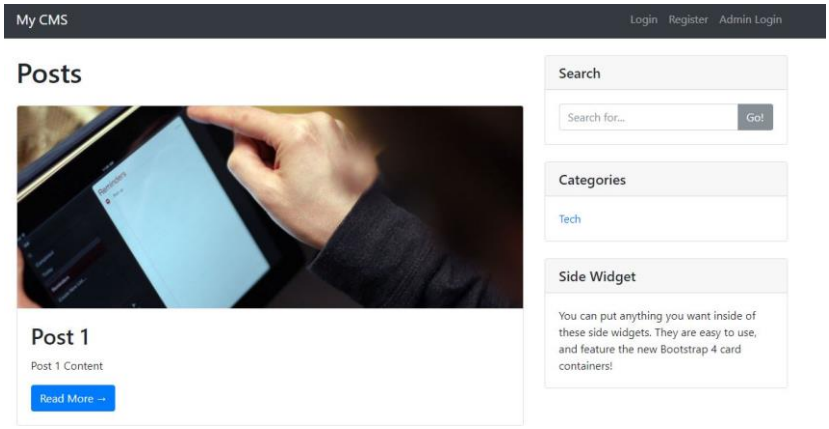
Αφού παρουσιάστηκε ο τρόπος εκτέλεσης τοπικά, στο επόμενο κεφάλαιο υπάρχουν αναλυτικές οδηγίες χρήσης, ξεχωριστά για τον κάθε ρόλο που αναπτύχθηκε.

Κεφάλαιο 7ο: Οδηγός Χρήσης

7.1 Για τον Επισκέπτη

7.1.1 Προβολή Άρθρων

Ο Επισκέπτης, μόλις μπαίνει στην εφαρμογή μπορεί να δει όλα τα άρθρα που έχουν δημιουργηθεί:

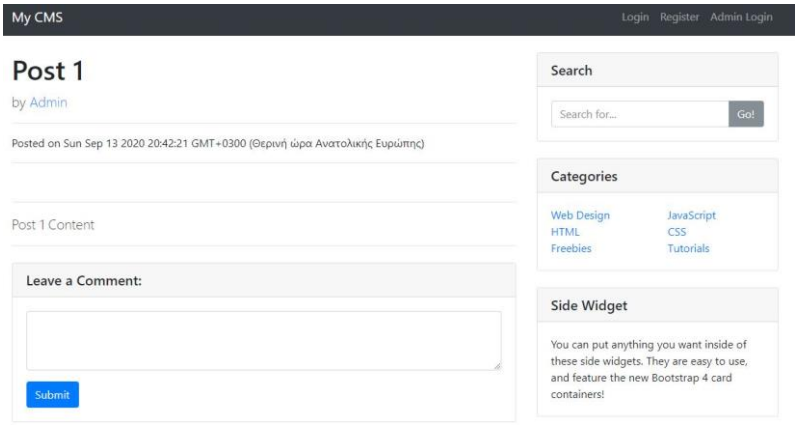


Εικόνα 12. Κεντρική Σελίδα

Επιλέγοντας Read More, μπορεί να διαβάσει το κείμενο του άρθρου και να αφήσει κάποιο σχόλιο

7.1.2 Σχολιασμός

Για να αφήσει ένα σχόλιο, ο επισκέπτης επιλέγει Read More σε ένα άρθρο, πληκτρολογεί το σχόλιό του στην ενότητα Leave a Comment και κάνει κλικ στο κουμπί Submit. Έπειτα το σχόλιό του θα πρέπει να εγκριθεί από τον διαχειριστή της εφαρμογής και στη συνέχεια θα εμφανιστεί κάτω από το αντίστοιχο άρθρο.

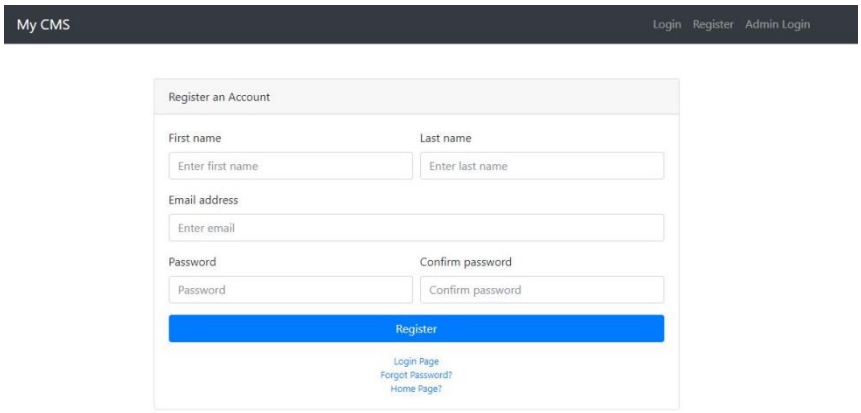


Εικόνα 13. Read More & Comments

7.2 Για τον Συγγραφέα

7.2.1 Register

Για την εγγραφή του στην πλατφόρμα, ένας μελλοντικός συγγραφέας επιλέγει Register:

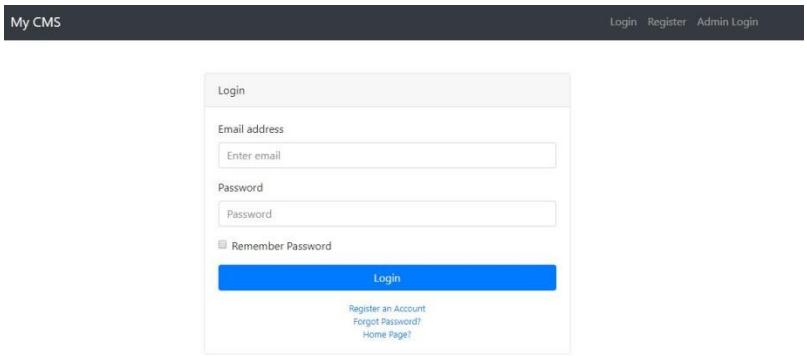


Εικόνα 14. Register form

Στη συνέχεια, εισάγει το Όνομά του, το Επώνυμο, μία email διεύθυνση και τον κωδικό πρόσβασης που θέλει να χρησιμοποιήσει και κάνει κλικ στο κουμπί Register.

7.2.2 Login

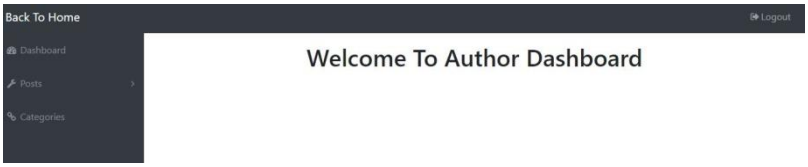
Αφού ολοκληρωθεί το registration, ο χρήστης μπορεί πλέον να κάνει Login στην εφαρμογή με τα στοιχεία σύνδεσης που έχει δημιουργήσει:



Εικόνα 15. Login form

7.2.3 Author Dashboard

Μόλις ένας συγγραφέας κάνει Login, έχει πρόσβαση στο dashboard συγγραφέα (Author Dashboard). Από εκεί μπορεί να δει όλα τα posts, να δημιουργήσει νέο και να επεξεργαστεί ένα υπάρχον. Μπορεί επίσης να δει όλες τις κατηγορίες, να δημιουργήσει νέα ή να επεξεργαστεί και να διαγράψει μία ήδη υπάρχουσα.



Εικόνα 16. Author Dashboard

7.2.4 Posts

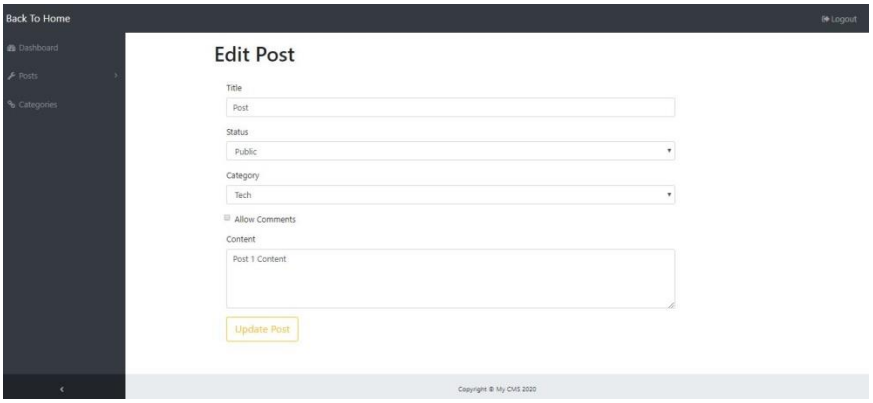
7.2.4.1 All Posts

Επιλέγοντας Posts και έπειτα All Posts, ένας συγγραφέας μπορεί να δει τα άρθρα που έχουν δημοσιευτεί.

Εικόνα 17. Author Dashboard – All Posts

7.2.4.2 Edit Posts

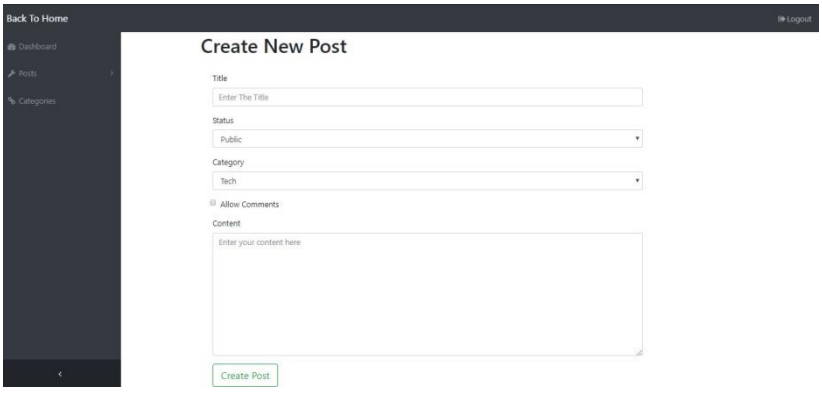
Επιλέγοντας Edit Post, μπορεί να επεξεργαστεί ένα άρθρο που έχει ήδη δημοσιευτεί και να το ενημερώσει επιλέγοντας Update Post



Εικόνα 18. Author Dashboard – Edit Post

7.2.4.3 Create New Post

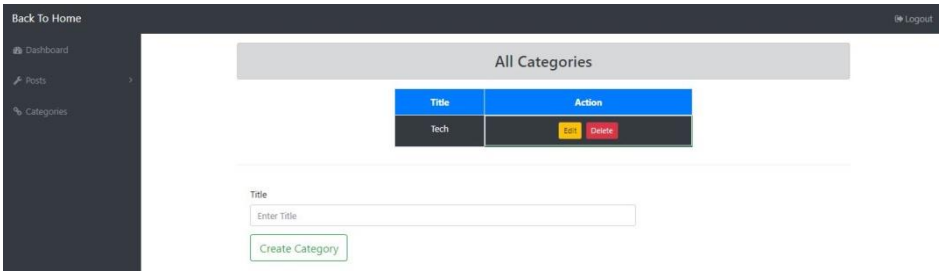
Επιλέγοντας Posts και έπειτα Create New Post, ένας συγγραφέας μπορεί να δημιουργήσει ένα νέο άρθρο. Για τη δημιουργία του άρθρου, θα πρέπει να πληκτρολογήσει τον Τίτλο του και να ορίσει εάν θα είναι Public, Private ή Draft. Τσεκάροντας την επιλογή Allow Comments, δίνεται η δυνατότητα στους επισκέπτες να σχολιάσουν το άρθρο αυτό. Έπειτα στο Content εισάγει το κείμενο του άρθρου και επιλέγοντας Create Post δημιουργεί το νέο άρθρο.



Εικόνα 19. Author Dashboard – Create New Post

7.2.5 Categories

Επιλέγοντας Categories, ένας συγγραφέας μπορεί να δει τις κατηγορίες που έχουν δημιουργηθεί, να μετονομάσει μία ήδη υπάρχουσα ή να διαγράψει μία κατηγορία. Μπορεί επίσης να δημιουργήσει μία νέα κατηγορία εισάγοντας τον Τίτλο της και κάνοντας κλικ στο κουμπί Create Category.



Εικόνα 20. Author Dashboard - Categories

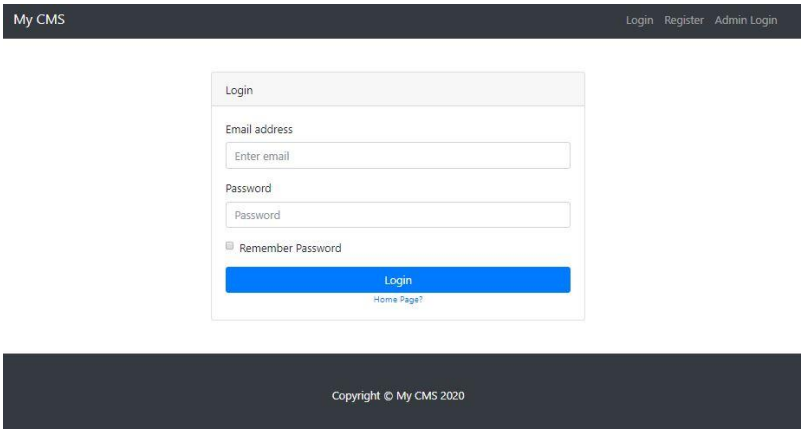
7.2.6 LogOut

Επιλέγοντας Logout, ένας συγγραφέας μπορεί αφού ολοκληρώσει τις αλλαγές του να αποσυνδεθεί από την εφαρμογή.

7.3 Για τον Διαχειριστή

7.3.1 Admin Login

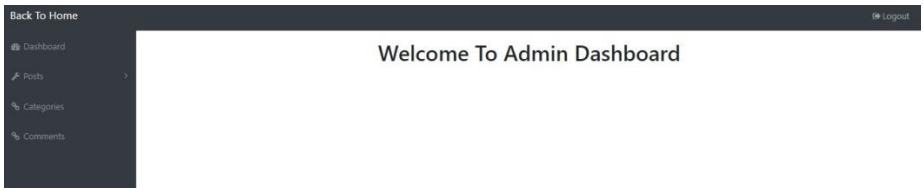
Για την είσοδό του στην εφαρμογή, ο διαχειριστής επιλέγει Admin Login και κάνει Login με τα ήδη υπάρχοντα στοιχεία σύνδεσής του.



Εικόνα 21. Admin Login

7.3.2 Admin Dashboard

Μετά τη σύνδεσή του στην εφαρμογή, ο διαχειριστής έχει πρόσβαση στο διαχειριστικό περιβάλλον από όπου μπορεί να να δει όλα τα posts, να δημιουργήσει νέο και να επεξεργαστεί ή να διαγράψει ένα υπάρχον. Μπορεί επίσης να δει όλες τις κατηγορίες, να δημιουργήσει νέα ή να επεξεργαστεί και να διαγράψει μία ήδη υπάρχουσα.



Εικόνα 22. Admin Dashboard

7.3.3 Posts

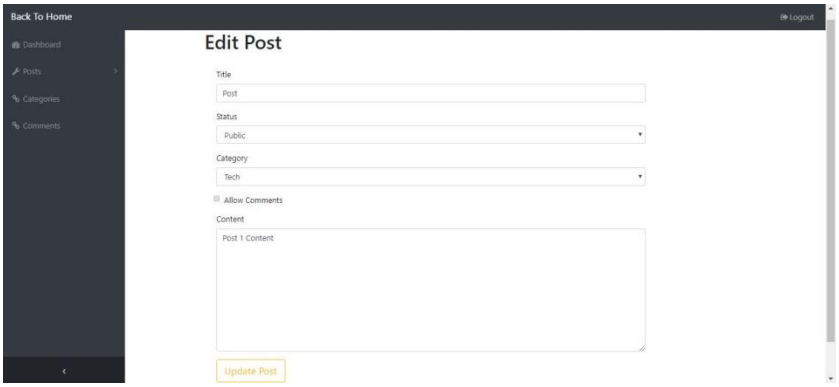
7.3.4 All Posts

Επιλέγοντας Posts και έπειτα All Posts, ένας συγγραφέας μπορεί να δει τα άρθρα που έχουν δημοσιευτεί.

Εικόνα 23. Admin Dashboard – All Posts

7.3.5 Edit Post

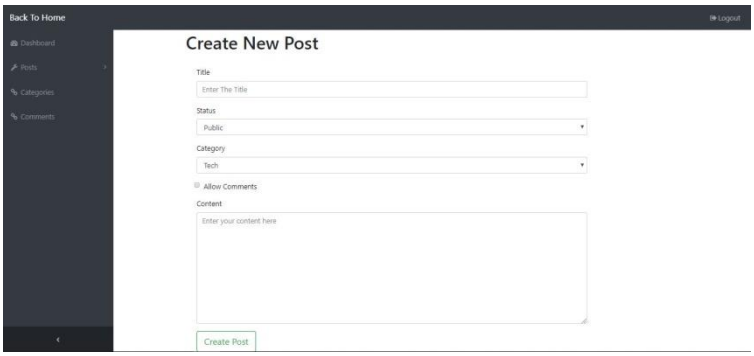
Επιλέγοντας Edit Post, μπορεί να επεξεργαστεί ένα άρθρο που έχει ήδη δημοσιευτεί και να το ενημερώσει επιλέγοντας Update Post.



Εικόνα 24. Admin Dashboard – Edit Post

7.3.6 Create Posts

Επιλέγοντας Posts και έπειτα Create New Post, ένας διαχειριστής μπορεί να δημιουργήσει ένα νέο άρθρο. Για τη δημιουργία του άρθρου, θα πρέπει να πληκτρολογήσει τον Τίτλο του και να ορίσει εάν θα είναι Public, Private ή Draft. Τσεκάροντας την επιλογή Allow Comments, δίνεται η δυνατότητα στους επισκέπτες να σχολιάσουν το άρθρο αυτό. Έπειτα στο Content εισάγει το κείμενο του άρθρου και επιλέγοντας Create Post δημιουργεί το νέο άρθρο.



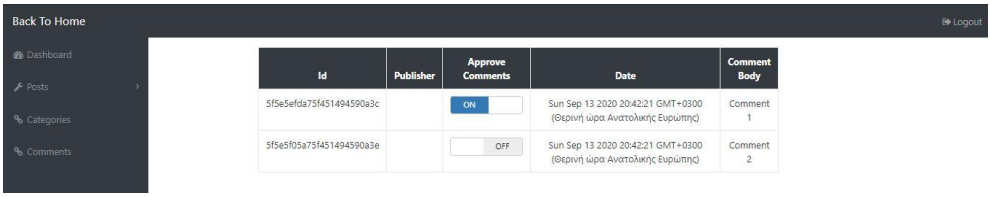
Εικόνα 25. Admin Dashboard – Create New Post

7.3.7 Delete Posts

Επιλέγοντας Delete, ο διαχειριστής μπορεί να διαγράψει ένα άρθρο.

7.3.8 Comments

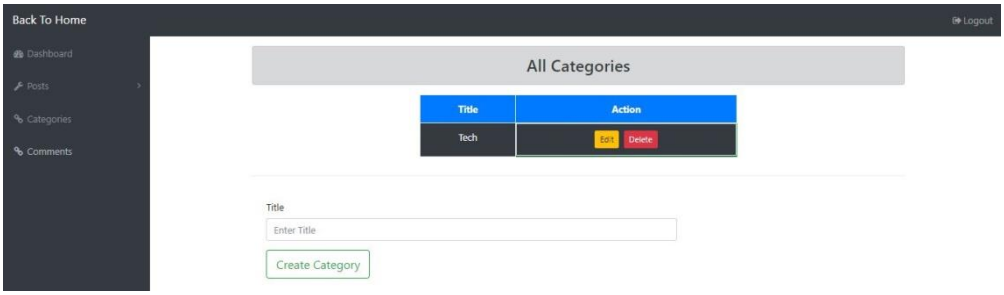
Επιλέγοντας Comments, ο διαχειριστής της εφαρμογής μπορεί να δει τα σχόλια που έχουν γίνει στα άρθρα. Επιλέγοντας On ή Off κάτω από τη στήλη Approve Comments, μπορεί να εγκρίνει ή όχι το κάθε σχόλιο.



Εικόνα 26. Admin Dashboard - Comments

7.3.9 Categories

Επιλέγοντας Categories, ένας διαχειριστής μπορεί να δει τις κατηγορίες που έχουν δημιουργηθεί, να μετονομάσει μία ήδη υπάρχουσα ή να διαγράψει μία κατηγορία. Μπορεί επίσης να δημιουργήσει μία νέα κατηγορία εισάγοντας τον Τίτλο της και κάνοντας κλικ στο κουμπί Create Category.



Εικόνα 27. Admin Dashboard - Categories

7.3.10 Logout

Επιλέγοντας Logout, ένας διαχειριστής μπορεί αφού ολοκληρώσει τις αλλαγές του να αποσυνδεθεί από την εφαρμογή.

7.4 Επίλογος

Στο κεφάλαιο αυτό παρουσιάστηκαν οι τρόποι χρήσης της κάθε λειτουργίας, ξεχωριστά για τον κάθε ρόλο χρήστη που έχει αναπτυχθεί. Στο επόμενο κεφάλαιο υπάρχουν τα συμπεράσματα συνολικά για την παρούσα εργασίες και τις τεχνικές που χρησιμοποιήθηκαν και οι τρόποι βελτίωσης αυτής.

Κεφάλαιο 8ο: Συμπεράσματα & Περιθώρια Βελτίωσης

8.1 Συμπεράσματα

Συμπερασματικά, στην εφαρμογή που αναπτύχθηκε προσομοιώνεται ένα CMS, στο οποίο υπάρχουν οι επισκέπτες, οι συγγραφείς και οι διαχειριστές, κατά την είσοδο των οποίων η εφαρμογή προσαρμόζεται αναλόγως σύμφωνα με τα αντίστοιχα δικαιώματα του κάθε ρόλου.

Γενικότερα η MongoDB είναι μία καλή και ευέλικτη πρακτική σε τέτοιου είδους projects, αλλά κυρίως σε μεγαλύτερα όπου η εγγραφές και τα queries στη βάση δεδομένων είναι περισσότερα μέσα σε πολύ μικρό χρονικό διάστημα. Σε κάθε περίπτωση, πριν να χρησιμοποιηθεί σε οποιοδήποτε project απαιτείται ανάλυση ώστε να διαπιστωθεί πότε και πώς θα πρέπει να χρησιμοποιηθεί, αλλιώς η χρήση της μπορεί να αποβεί μοιραία. Τέτοιου είδους περιπτώσεις είναι η μη επαρκής μνήμη και τοπικός αποθηκευτικός χώρος, όπως επίσης η ύπαρξη πολλών συσχετίσεων μεταξύ των πινάκων.

8.2 Περιθώρια Βελτίωσης

Ένα σημαντικό βήμα για τη βελτίωση της παρούσας εφαρμογής θα ήταν ο διαχειριστής να έχει τη δυνατότητα να ελέγχει ποιος γίνεται author, να εγκρίνει πρώτα το αίτημά του και έπειτα ο κάθε author να μπορεί να εισέλθει στην πλατφόρμα. Σχετικά με τους ρόλους χρηστών, θα ήταν επίσης καλό ένας διαχειριστής να μπορεί να ορίσει πότε ένας author θα είναι και διαχειριστής. Στην κατάσταση που είναι αυτή τη στιγμή η εφαρμογή, πρακτικά μπορεί οποιοσδήποτε να γίνει διαχειριστής, αρκεί να επιλέξει Admin Login και να χρησιμοποιήσει τα ήδη υπάρχοντα στοιχεία σύνδεσής του, γεγονός το οποίο αποτελεί θέμα ασφαλείας.

Επιπρόσθετα, ένα ακόμα σημείο που μπορεί να βελτιστοποιηθεί είναι τα warning messages, τα οποία σε πολλά σημεία δεν υπάρχουν. Για παράδειγμα, όταν ένας διαχειριστής επιλέξει να διαγράψει ένα άρθρο ή μία κατηγορία, θα μπορούσε να εμφανίζεται ένα μήνυμα που να τον ρωτάει εάν είναι σίγουρος ότι επιθυμεί να προχωρήσει σε διαγραφή. Με αυτό τον τρόπο, μπορούν να αποτραπούν οι διαγραφές εκ παραδρομής.

Σαν μία επιπλέον προσθήκη στις λειτουργίες της εφαρμογής, θα μπορούσε να προστεθεί το auditing. Εφόσον η MongoDB παρέχει και υποστηρίζει logs και auditing, θα μπορούσαν να χρησιμοποιηθούν από την πλευρά του διαχειριστή, ώστε να καταγράφονται οι ενέργειες του κάθε χρήστη και η οποιαδήποτε μεταβολή στη βάση δεδομένων. Εκτός από επιπλέον δικλείδα ασφαλείας, οι συγκεκριμένες λειτουργίες μπορούν να βοηθήσουν και στη βελτιστοποίηση της απόδοσης της εφαρμογής.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Internet Sites

[1] https://en.wikipedia.org/wiki/CAP_theorem

[2] https://geeknarrator.com/2017/10/02/cap-theorem-consistency-availability-and-partition-tolerance/

[5] https://en.wikipedia.org/wiki/NoSQL

[7] https://en.wikipedia.org/wiki/Document-oriented_database

[8] https://www.dataversity.net/understanding-key-value-databases/#

[9] https://en.wikipedia.org/wiki/Key-value_database

[10] https://docs.mongodb.com/manual/

[11] https://docs.mongodb.com/manual/crud/

[12] https://docs.mongodb.com/manual/aggregation/

[13] https://docs.mongodb.com/manual/replication/

[14] https://docs.mongodb.com/manual/sharding/

[17] http://pacific.jour.auth.gr/content_management_systems/

[18] https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs

[19] https://www.tutorialspoint.com/mongodb/mongodb_environment.htm

Journal Articles

[3] Tsuyuzaki, K., Onizuka, M. (2012). NoSQL Database Characteristics and Benchmark System. NTT Technical Review.

[4] Harpreet kaur, Jaspreet kaur, Kamaljit kaur. "A Review of Non Relational Databases, Their Types, Advantages And Disadvantages", International Journal of Engineering Research & Technology, Vol. 2 Issue 2 February 2013

[6] Nikhil Dasharath Karande "A Survey Paper on NoSQL Databases: Key-Value Data Stores and Document Stores"**,** International Journal of Research in Advent Technology, Vol.6, No.2, February 2018

[15] Neal Leavitt " Will NoSQL Databases Live Up to their Promise?" IEEE Computer Society 2010

Papers in Conference Proceedings

[16]Lior Okman, Nurit Gal-Oz, Yaron Gonen, Ehud Gudes, Jenny Abramov (2011), "Security Issues In NoSQL Dtabases", International Joint Conference of IEEE