ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη πρότυπης εφαρμογής διαδικτύου με τη χρήση σύγχρονων τεχνολογιών εφαρμογών διαδικτύου και ανάλυσης δεδομένων μεγάλου όγκου

Ανδρέας - Π - Χριστόπουλος

Επιβλέποντες : **Ιωάννης Χαμόδρακας,** ΕΔΙΠ Τμήματος Πληροφορικής και Τηλεπικοινωνιών
Αφροδίτη Τσαλγατίδου, Αναπληρώτρια Καθηγήτρια

ΑΘΗΝΑ

ΙΑΝΟΥΑΡΙΟΣ 2017

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη πρότυπης εφαρμογής διαδικτύου με τη χρήση σύγχρονων τεχνολογιών εφαρμογών διαδικτύου και ανάλυσης δεδομένων μεγάλου όγκου

Ανδρέας Π. Χριστόπουλος

Α.Μ.: 1115201000123

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Ιωάννης Χαμόδρακας,** ΕΔΙΠ Τμήματος Πληροφορικής και Τηλεπικοινωνιών
Αφροδίτη Τσαλγατίδου, Αναπληρώτρια Καθηγήτρια

ΠΕΡΙΛΗΨΗ

Με την πάροδο των χρόνων, οι διαδικτυακές εφαρμογές γνωρίζουν μεγάλες και διαδοχικές αλλαγές τόσο ως προς τις νέες τεχνολογίες που συνεχώς εισέρχονται όσο και ως προς τις απαιτήσεις που υπάρχουν από τους χρήστες του διαδικτύου. Ο όγκος των δεδομένων που καλούνται να διαχειριστούν οι σύγχρονες εφαρμογές διαδικτύου αυξάνεται συνεχώς και με πολύ γρήγορους ρυθμούς, ενώ ταυτόχρονα τα δεδομένα αυτά, τα Μεγάλα Δεδομένα, είναι ακατόρθωτο να ομαδοποιηθούν λόγω του όγκου και της φύσης τους (σύγχρονες εφαρμογές έρχονται αντιμέτωπες με διαφορετικού τύπου δεδομένα – εικόνες, βίντεο, δημοσιεύσεις, ετικέτες, κλπ). Κάτι τέτοιο γεννά την ανάγκη να εισαχθούν νέες τεχνολογίες που θα βοηθήσουν τους σύγχρονους προγραμματιστές να ανταπεξέλθουν στις απαιτήσεις της αγοράς. Τόσο οι προγραμματιστές στον τομέα των διαδικτυακών εφαρμογών όσο και στον τομέα της ανάλυσης δεδομένων είναι σε θέση πλέον να επιλέξουν μέσα από μια πληθώρα πλαισίων λογισμικού τις τεχνολογίες αυτές που θα τους διευκολύνουν στη δουλειά τους και θα καταφέρουν να πετύχουν τα επιθυμητά αποτελέσματα με τον πλέον σύγχρονο και αποδοτικό τρόπο.

Σκοπός της παρούσας εργασίας είναι η μελέτη σύγχρονων τεχνολογιών που θα βοηθήσουν στην ανάπτυξη μιας σύγχρονης εφαρμογής διαδικτύου που έρχεται αντιμέτωπη με Μεγάλα Δεδομένα καθώς και η ανάπτυξη ενός παραδείγματος που θα παρουσιάζει τις τεχνολογίες που θα μελετηθούν (βλ. Παράρτημα 2 – Παρουσίαση της διαδικτυακής εφαρμογής - παραδείγματος). Η μελέτη τεχνολογιών θα επεκταθεί και στο τμήμα της ανάλυσης Μεγάλων Δεδομένων, καθώς θα αναπτυχθεί και ένα σύστημα προτάσεων για τη διαδικτυακή εφαρμογή (βλ. Παράρτημα 1 - Αρχιτεκτονική της εφαρμογής).

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Τεχνολογίες Εφαρμογών διαδικτύου και σύστημα προτάσεων - διαχείρισης Μεγάλων Δεδομένων

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Τεχνολογίες Εφαρμογών Διαδικτύου, Διαδίκτυο, Πρόβλεψη Βαθμολογιών, Ομοιότητες Χρηστών, Μετρικές Αξιολόγησης, Μεγάλα Δεδομένα

ABSTRACT

As time goes by, web applications become more and more complicated when it comes to newer technologies that appear day after day and the expectations of the internet users change quicker than ever. The size of the data that an application has to deal with grows very fast and so does the complexity. These Big Data are large volumes of data that cannot be structured easily or cannot be structured at all. In order to overcome the problems with the amount of data and their complexity, new technologies have been developed to deal with the rising expectations of the users around the world. Web developers as well as data analysts can now choose among a great variety of frameworks that can help them build their applications and their code easily and as efficiently as possible.

The purpose of this thesis is to study these new technologies that will help a web developer build an efficient web application that uses Big Data and to develop an example of a web application using the technologies that will have been studied. Moreover, technologies for Big Data analysis will be studied to develop a recommender system for the web application.

SUBJECT AREA: Web Applications Technologies and Recommender System – processing on Big Data

KEYWORDS: Web Applications Technologies, Web, Personalization, Collaborative Filtering, Rating Estimation Similarities, Evaluation Metrics, Big Data

Η παρούσα πτυχιακή εργασία είναι αφιερωμένη στην οικογένεια μου και σε όλους τους φίλους που με στήριξαν στην εκπόνησή της

ΕΥΧΑΡΙΣΤΙΕΣ

Για τη διεκπεραίωση της παρούσας Πτυχιακής Εργασίας, θα ήθελα να ευχαριστήσω τους επιβλέποντες, κ. Ιωάννη Χαμόδρακα, κα Αφροδίτη Τσαλγατίδου για τη συνεργασία και την πολύτιμη συμβολή τους στην ολοκλήρωση της, για την εμπιστοσύνη που μου έδειξαν και για τους ορίζοντες που μου ανοίχθηκαν κατά την ενασχόλησή μου με τη θεματική ενότητα που μου ανέθεσαν να μελετήσω.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΠΡΟΛΟΓΟΣ

Η παρούσα πτυχιακή εργασία πραγματοποιήθηκε στο Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών στο τμήμα Πληροφορικής και Τηλεπικοινωνιών. Στόχος της εργασίας είναι η μελέτη και εφαρμογή νέων τεχνολογιών για την ανάπτυξη μιας σύγχρονης διαδικτυακής εφαρμογής. Ταυτόχρονα θα υπάρξει ανάπτυξη ενός παραδείγματος διαδικτυακής εφαρμογής με χρήση των τεχνολογιών που συμπεριλαμβάνονται στη μελέτη, καθώς επίσης και ενός συστήματος προτάσεων με τη χρήση τεχνολογιών για ανάλυση δεδομένων τεράστιου όγκου.

1. ΕΙΣΑΓΩΓΗ

1.1 Τι είναι τα "Μεγάλα Δεδομένα"

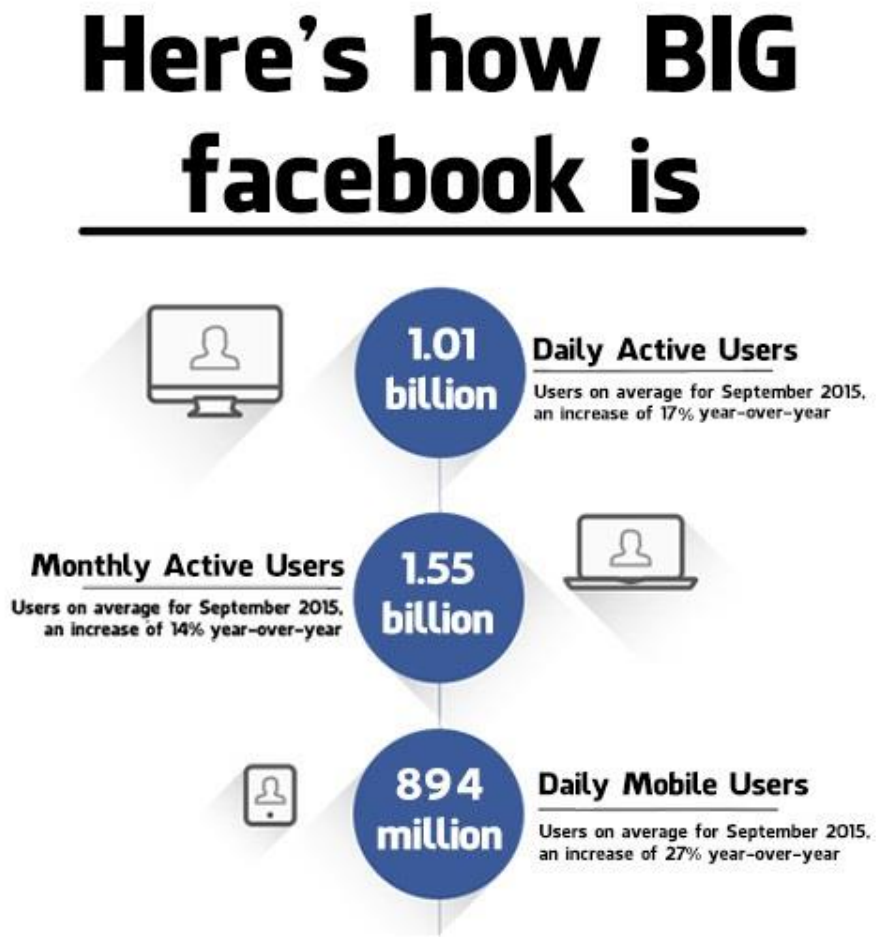
Με τον όρο Μεγάλα Δεδομένα (Big Data) ονομάζουμε τα σύνολα δεδομένων που είναι πολύ μεγάλα και πολύ πολύπλοκα για να αντιμετωπιστούν με παραδοσιακές μεθόδους και λύσεις. Η πολυπλοκότητα των Μεγάλων Δεδομένων έχει να κάνει με τη δυσκολία που παρουσιάζεται στο να μπορέσει κανείς να τα κατηγοριοποιήσει, επομένως πολλές φορές παρατηρείται ότι τα δεδομένα αυτά είναι μη ομαδοποιήσιμα - αδόμητα (unstructured data) [1].

Με την πάροδο των χρόνων τα δεδομένα στο διαδίκτυο αυξάνονται με ραγδαίους ρυθμούς. Κατά μέσο όρο, ανά 18 μήνες τα ήδη υπάρχοντα δεδομένα διπλασιάζονται σε όγκο και σε πολυπλοκότητα. Κάτι τέτοιο μας βρίσκει αντιμέτωπους με έναν συνεχώς αυξανόμενο τεράστιο όγκο δεδομένων που μάλιστα γίνονται ολοένα και πιο αδόμητα [2].

Γίνεται λοιπόν εύκολα αντιληπτό ότι η διαχείριση τεράστιου όγκου αδόμητων δεδομένων με συμβατικές μεθόδους καθίσταται από δύσκολη και χρονοβόρα έως αδύνατη διαδικασία. Άρα, νέα συστήματα κάνουν την εμφάνιση τους, τόσο από άποψη υλικού, όσο και λογισμικού αλλά και, το σημαντικότερο, έξυπνης αντιμετώπισης και υλοποίησης ώστε να αντιμετωπίζουν για παράδειγμα τμηματικά τον όγκο των δεδομένων, εστιάζοντας σε υποτμήματά του.

1.1.1 Παράδειγμα Μεγάλων Δεδομένων του Facebook

Για να το δούμε πρακτικά, ας πάρουμε σαν παράδειγμα τα δεδομένα του κοινωνικού δικτύου Facebook:



Εικόνα 1: Τα δεδομένα του Facebook για το έτος 2015 [16]

Βλέπουμε λοιπόν έναν μεγάλο αριθμό χρηστών να συνδέονται μάλιστα μέσω διαφορετικών πλατφορμών (διαδίκτυο, έξυπνες φορητές συσκευές κλπ). Αν αναλογιστούμε τα διαφορετικά δεδομένα που πρέπει να αποθηκεύονται για κάθε χρήστη (φωτογραφίες, μηνύματα, likes, ετικέτες κλπ), φτάνουμε εύκολα στο συμπέρασμα ότι η κεντρική βάση δεδομένων του facebook αποτελείται από έναν τεράστιο όγκο μη ομαδοποιήσιμων δεδομένων, συνεχώς αυξανόμενο.

Προφανώς τα δεδομένα του facebook ανήκουν στην κατηγορία των Μεγάλων Δεδομένων. Ο όγκος των δεδομένων του facebook είναι τώρα πια της τάξης των εκατοντάδων PentaBytes - PB (1 PB = 1.000.000.000.000.000 B = 10^{15} bytes = 1.000 terabytes).

Είναι σαφές ότι με έναν τόσο μεγάλο όγκο από μη ομαδοποιήσιμα δεδομένα, η παραγωγή μετρικών και αναλυτικών στοιχείων γίνεται πολύ περίπλοκη και οπωσδήποτε πιο χρονοβόρα διαδικασία. Έτσι λοιπόν, καταφεύγουμε σε νέες τεχνικές για τον σχεδιασμό της κεντρικής βάσης της εφαρμογής, και σε νέες τεχνολογίες για τη διαχείριση των αδόμητων δεδομένων της.

2. ΣΧΕΣΙΑΚΕΣ ΚΑΙ ΜΗ ΣΧΕΣΙΑΚΕΣ ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ ΓΙΑ ΤΗΝ ΥΠΟΣΤΗΡΙΞΗ ΜΕΓΑΛΩΝ ΔΕΔΟΜΕΝΩΝ

2.1 Σχεσιακή βάση δεδομένων

Οι Σχεσιακές Βάσεις Δεδομένων (Relational Databases) είναι ικανές και προτιμούνται για την υποστήριξη μικρού έως μεσαίου όγκου δεδομένων και πάνω απ' όλα δεδομένων που μπορούν να ομαδοποιηθούν εύκολα. Όταν όμως ο όγκος των δεδομένων είναι τεράστιος και μάλιστα συνεχώς αυξάνει, οι δυνατότητες μιας σχεσιακής βάσης να τον στηρίξουν μειώνονται σημαντικά.

Η μείωση των δυνατοτήτων της σχεσιακής βάσης δεν οφείλεται μόνο στον όγκο των δεδομένων. Οφείλεται σε σημαντικό βαθμό και στο φαινόμενο των αδόμητων δεδομένων που αναφέρεται παραπάνω. Οι σχεσιακές βάσεις δεν είναι προορισμένες για τη διαχείριση τέτοιου είδους δεδομένων.

Γίνεται σαφές λοιπόν ότι για τον σχεδιασμό της βάσης δεδομένων μιας εφαρμογής που διαχειρίζεται αδόμητα δεδομένα και μάλιστα τεραστίου μεγέθους, νέες τεχνολογίες θα πρέπει να εισαχθούν όσον αφορά τον σχεδιασμό και τη συντήρηση της βάσης.

2.2 Μη Σχεσιακή βάση δεδομένων

Μια σύγχρονη τεχνολογία για τη σχεδίαση και χρήση βάσεων δεδομένων είναι ο θεσμός των Μη Σχεσιακών Βάσεων Δεδομένων (Non Relational Databases). Αξίζει να σημειωθεί ότι οι τεχνολογία των βάσεων δεδομένων τέτοιου τύπου ξεκίνησε να αναπτύσσεται από τις αρχές του 2000 από διάφορους φορείς και σήμερα έχουμε καταλήξει με μια πληθώρα από τεχνικές σχεδίασης βάσεων δεδομένων και σύγχρονων πλαισίων λογισμικού (frameworks) που δουλεύουν πάνω σε αυτές.

Υπάρχουν διάφοροι τύποι Μη Σχεσιακών Βάσεων Δεδομένων (key-value store, αντικειμενοστραφείς, αρχειοστραφείς κλπ). [3]

2.3 Σχεσιακή σε σύγκριση με Μη Σχεσιακή Βάση Δεδομένων

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.

Οι Μη Σχεσιακές Βάσεις υπερτερούν σε διάφορους τομείς, όπως στους ακόλουθους, και για το λόγο αυτό έχουν ξεκινήσει να εισάγονται στις ζωές ολοένα και περισσότερων προγραμματιστών [4].

Πιο αναλυτικά για τα πλεονεκτήματα των Μη Σχεσιακών Βάσεων Δεδομένων έναντι των Σχεσιακών σε ότι αφορά τις διαδικτυακές κυρίως εφαρμογές:

- Οι Μη Σχεσιακές Βάσεις Δεδομένων είναι πιο δυναμικές
- Είναι πιο επεκτάσιμες (scalable) από τις Σχεσιακές
- Το κόστος είναι μικρότερο

2.3.1 Δυναμικότητα

Τα δεδομένα του διαδικτύου, όπως έχει προαναφερθεί, 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.

Πιο συγκεκριμένα, 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.

2.1.1 Επεκτασιμότητα

Τα δεδομένα μιας εφαρμογής που είναι διαθέσιμα δημοσίως, προσπελαύνονται από χρήστες ανά τον κόσμο σε διαφορετικές γλώσσες και από όλο και περισσότερες συσκευές. Η επεκτασιμότητα στις Σχεσιακές Βάσεις είναι ένα στοιχείο που λείπει και είναι καθήκον του προγραμματιστή να φροντίσει γι’ αυτό.

Αντιθέτως, οι Μη Σχεσιακές Βάσεις Δεδομένων έχουν αναπτυχθεί ώστε να είναι επεκτάσιμες, οπότε μια εφαρμογή μπορεί να επεκτείνει τα δεδομένα που χρησιμοποιεί και τους πόρους που χρειάζεται και η βάση δεδομένων να αντιμετωπίσει μια τέτοιου είδους κατάσταση με τρόπο αυτοματοποιημένο.

2.1.2 Κόστος

Για τον σχεδιασμό και τη συντήρηση μιας Σχεσιακής Βάσης Δεδομένων χρειάζεται να υφίσταται ένα μεγάλο τμήμα ανθρωπίνου δυναμικού που θα πρέπει συνεχώς να εργάζεται για την ομαλή λειτουργία της βάσης και για τις ενδεχόμενες αλλαγές και βελτιστοποιήσεις που μπορεί να χρειαστούν. Ακόμη, πολλές φορές μπορεί να χρειάζονται νέοι πόροι (τερματικά, διαφορετικά συστήματα αρχείων κλπ) που να απαιτούνται για τη στήριξη μιας Σχεσιακής Βάσης όσο οι απαιτήσεις μιας εφαρμογής αυξάνονται.

Από την άλλη πλευρά, όταν μια εφαρμογή στηρίζεται σε μια Μη Σχεσιακή Βάση Δεδομένων, μια μικρότερη παραγωγική ομάδα ανθρώπων, μπορεί να σχεδιάσει και να συντηρήσει ευκολότερα και οικονομικότερα τη βάση δεδομένων και να επιτύχει έως και 90% οικονομικότερα τις εργασίες που προβλέπονται για μια βάση δεδομένων.

2.4 Παρουσίαση Αρχειοστραφούς Μη Σχεσιακής Βάσης Δεδομένων

Σε μια αρχειοστραφή (document oriented) ένα αρχείο (document) είναι η βασική μονάδα δεδομένων. Ας το παρομοιάσουμε με μια πλειάδα (row) σε μια σχεσιακή βάση.

Ένα παράδειγμα αρχείου από μια αρχειοστραφή βάση δεδομένων φαίνεται στην ακόλουθη εικόνα:

Εικόνα 2: Αρχείο αποθηκευμένο σε αρχειοστραφή βάση δεδομένων 1 [17]

Βλεπουμε λοιπόν μια «πλειάδα» ενός βιβλίου σε κωδικοποίηση JSON (Javascript Object Notation). Να σημειωθεί ότι συνήθης τεχνική αποθήκευσης αρχείων σε μορφή JSON μέσα σε μια αρχειοστραφή βάση δεδομένων είναι η σειριοποίησή τους σε μορφή BSON (Binary Structured Object Notation). Η σειριοποίηση σε μορφή BSON παρουσιάζεται παρακάτω πιο αναλυτικά.

Ας δούμε ένα πιο περίπλοκο παράδειγμα αρχείου σε μορφή JSON. Έστω ότι έχουμε στη βάση μας έναν κατάλογο προϊόντων διαφορετικών μεταξύ τους. Έστω ότι τα προϊόντα είναι 2. Το εν λόγω αρχείο λοιπόν θα είναι της μορφής

Εικόνα 3: Αρχείο αποθηκευμένο σε αρχειοστραφή βάση δεδομένων 2 [17]

Συμπεραίνουμε λοιπόν ότι έχουμε μια πιο χαλαρή δομή, δηλαδή δεν είναι απαραίτητο τα «πεδία μιας όψης» (με ορολογία σχεσιακής βάσης) να είναι όμοια.

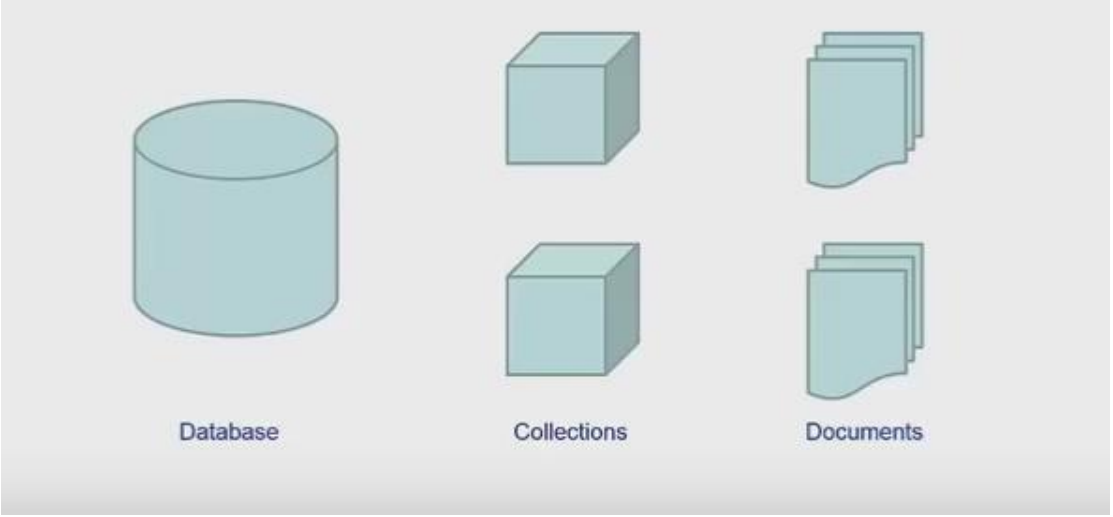
Ακόμη ένα παράδειγμα αρχείου:

Εικόνα 4: Αρχείο αποθηκευμένο σε αρχειοστραφή βάση δεδομένων 3 [17]

Όπως βλέπουμε, το πεδίο Author περιέχει ένα εμφωλευμένο αντικείμενο (nested object) για να περιγράψει τους 2 συγγραφείς που έχει το βιβλίο του παραδείγματος. Επομένως συμπεραίνουμε ότι σε μια αρχειστραφή βάση δεδομένων μπορούμε να έχουμε πολύ πιο περίπλοκες «πλειάδες».

Η δομή μιας αρχειοστραφούς βάσης αποτελείται από συλλογές δεδομένων (collections) σαν αυτά που παρουσιάζονται στις παραπάνω εικόνες. Χρησιμοποιούνται συλλογές ώστε να υπάρχει μια στοιχειώδης κατηγοριοποίηση των δεδομένων με σκοπό την ευχρηστία.

Στην παρακάτω εικόνα φαίνεται γραφικά η δομή μιας αρχειοστραφούς βάσης



Εικόνα 5: Δομή αρχειοστραφούς βάσης δεδομένων [17]

Μια τέτοιου είδους δομή της βάσης δεδομένων προσφέρει ευκολία σε πιθανές αλλαγές του σχήματος της βάσης (είναι αρκετά εύπλαστη δομή) και ελευθερία στον τρόπο που θα επιλέξει ο χρήστης να ομαδοποιήσει τα αποθηκευμένα δεδομένα, ανάλογα με τις ανάγκες της εφαρμογής. Επομένως είναι μια καλή λύση για εφαρμογές που έχουν να κάνουν με μη ομαδοποιήσιμα Μεγάλα Δεδομένα.

2.5 Βασικό συμπέρασμα

Όταν μια εφαρμογή έρχεται αντιμέτωπη με Μεγάλα Δεδομένα, μια σχεσιακή βάση δεδομένων δεν μπορεί πλέον να ανταπεξέλθει στις απαιτήσεις των χρηστών και στις προσδοκίες των προγραμματιστών που αναπτύσσουν την εφαρμογή κατ' επέκταση. Για τον λόγο αυτό, η βάση δεδομένων μιας εφαρμογής που διαχειρίζεται Μεγάλα Δεδομένα θα πρέπει να στηρίζεται από μια Μη Σχεσιακή Βάση Δεδομένων.

3. ΥΠΗΡΕΣΙΟΣΤΡΑΦΗΣ ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΚΑΙ RESTFUL ΔΙΑΔΙΚΤΥΑΚΕΣ ΥΠΗΡΕΣΙΕΣ

Όταν μιλάμε για υπηρεσιοστραφή αρχιτεκτονική (SOA) στις διαδικτυακές εφαρμογές, εννοούμε ένα σύνολο από διαδικτυακές υπηρεσίες που αλληλεπιδρούν μεταξύ τους, είτε ανταλλάσσοντας πληροφορία είτε πόρους ώστε να εκτελεστεί μια συγκεκριμένη διεργασία [5].

Υπάρχουν διάφορα πρωτόκολλα που λειτουργούν βάσει της υπηρεσιοστραφούς αρχιτεκτονικής, καθένα με τα δικά του χαρακτηριστικά, όπως το SOAP, το WSDL ή το REST το οποίο και θα παρουσιαστεί αναλυτικότερα στην παρούσα μελέτη.

Σχεδόν όλες οι σύγχρονες διαδικτυακές εφαρμογές έχουν εισάγει στη λειτουργία τους (ή αρχίζουν σιγά - σιγά να εισάγουν) την έννοια των RESTful διαδικτυακών Υπηρεσιών (Web Services) έναντι των SOAP Διαδικτυακών Υπηρεσιών που μέχρι πρότινος ήταν αυτά που ευρέως χρησιμοποιούνταν.
Η διαδικασία της εισαγωγής RESTful Web Services δεν πρόκειται για κάποια «νέα μόδα» στον τομέα των διαδικτυακών εφαρμογών.

Τα πλεονεκτήματα των RESTful διαδικτυακών Υπηρεσιών γίνονται πιο ξεκάθαρα αν αναλογιστούμε ότι το REST χρησιμοποιεί απλά http αιτήματα για την επικοινωνία Εξυπηρετητή – Πελάτη. Κάτι τέτοιο κάνει αρκετά πιο αποδοτική και ευκολότερα επεκτάσιμη μια διαδικτυακή εφαρμογή που χρησιμοποιεί RESTful Διαδικτυακές Υπηρεσίες.

3.1 Υπηρεσιοστραφής Αρχιτεκτονική

Πιο συγκεκριμένα σε ότι αφορά την έννοια της Υπηρεσιοστραφούς Αρχιτεκτονικής, θα μπορούσαμε να την παρουσιάσουμε ως ένα σύνολο διαδικτυακών υπηρεσιών – μη ισχυρά συνδεδεμένων μεταξύ τους που λειτουργούν ανεξάρτητα και μπορούν να επιτύχουν το επιθυμητό αποτέλεσμα χωρίς να απαιτείται οι υπηρεσίες αυτές να «γνωρίζονται μεταξύ τους».

Βασικό χαρακτηριστικό μιας υπηρεσιοστρεφούς αρχιτεκτονικής είναι η υποστήριξή της από υπηρεσίες οι οποίες οφείλουν να είναι σαφώς καθορισμένες, αυτόνομες, και μην εξαρτώνται από το πλαίσιο ή την κατάσταση των άλλων υπηρεσιών. Κάθε τέτοια υπηρεσία αποτελεί μια μονάδα εργασίας που γίνεται διαθέσιμη από έναν πάροχο (Server), ώστε να παρέχει τα επιθυμητά τελικά αποτελέσματα στον καταναλωτή (Client) υπηρεσίας (πχ από ένα πρόγραμμα περιήγησης).

Συμπεραίνουμε λοιπόν ότι για να χρησιμοποιηθεί μια υπηρεσιοστραφής αρχιτεκτονική για το σύστημα υποστήριξης (backend) μιας εφαρμογής θα πρέπει να λαμβάνονται υπ’ όψιν σχεδιαστικές αρχές και τεχνολογίες που μια τέτοιου είδους δομή απαιτεί.

3.2 Restful Διαδικτυακές Υπηρεσίες

Ο όρος REST προέρχεται από τα αρχικά του «**Re**presentational **S**tate **T**ransfer». Πρόκειται για ένα stateless πρωτόκολλο επικοινωνίας μεταξύ προγραμμάτων πελάτη

(client) και εξυπηρετητή (server) – επικοινωνία που εδραιώνεται με τη χρήση αιτημάτων βάσει του πρωτοκόλλου HTTP [6].

Ουσιαστικά, το REST είναι μια αρχιτεκτονική με βάση την οποία σχεδιάζονται διαδικτυακές εφαρμογές. Το βασικό κριτήριο της αρχιτεκτονικής αυτής είναι να χρησιμοποιούνται απλά αιτήματα HTTP και όχι περίπλοκοι μηχανισμοί ώστε να επιτευχθεί η επικοινωνία μεταξύ 2 ή περισσότερων μηχανημάτων. Το ίδιο το διαδίκτυο γενικώς είναι βασισμένο σε HTTP αιτήματα, επομένως μπορεί και αυτό να θεωρηθεί σαν μια αρχιτεκτονική REST για την επικοινωνία ενός προγράμματος πελάτη (ας πούμε ενός φυλλομετρητή σαν ένα κλασικό, καθημερινό παράδειγμα).

Οι RESTful εφαρμογές χρησιμοποιούν τα HTTP αιτήματα για την εισαγωγή – δημιουργία – ενημέρωση (POST και PUT), το διάβασμα – την προσπέλαση (GET) και τη διαγραφή δεδομένων (DELETE). Επομένως, το REST χρησιμοποιεί όλα τα HTTP ρήματα (HTTP verbs) για τις βασικές CRUD (Create/Read/Update/Delete) διαδικασίες.

Η αρχιτεκτονική REST είναι σίγουρα μια πιο «ελαφριά» εναλλακτική για μηχανισμούς τύπου Remote Procedure Calls και για διαδικτυακές υπηρεσίες (SOAP, WSDL, κλπ.). Αν και αρκετά απλό, μέσω του REST μπορεί να πραγματοποιηθεί οτιδήποτε μπορεί να γίνει με οποιαδήποτε διαδικτυακή υπηρεσία.

Η αρχιτεκτονική REST δεν πρόκειται για μια στατική και στάνταρ αρχιτεκτονική. Μπορεί να σχεδιαστεί βάσει των αναγκών της εφαρμογής που καλείται να υποστηρίξει. Επιπλέον, ολοένα και περισσότερες γλώσσες προγραμματισμού παρέχουν σύγχρονα πλαίσια λογισμικού για τη σχεδίαση RESTfull εφαρμογών.

4. ΜΙΑ ΔΙΑΔΙΚΤΥΑΚΗ ΕΦΑΡΜΟΓΗ ΣΤΟ ΜΟΝΤΕΛΟ JAVASCRITP – REST API ΜΕ ΧΡΗΣΗ ΤΗΣ MONGODB ΩΣ ΚΕΝΤΡΙΚΗ ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

Με την πάροδο των χρόνων, ο προγραμματισμός του εμπρόσθιου τμήματος (front end) μιας διαδικτυακής εφαρμογής έχει γνωρίσει μια ραγδαία ανάπτυξη σε κάθε τομέα του. Νέες τεχνολογίες ξεκίνησαν χρόνο με το χρόνο να εμφανίζονται συμπληρώνοντας η μία τις λειτουργίες της άλλης.

Ένας από τους κυριότερους τομείς του προγραμματισμού του εμπρόσθιου τμήματος που συνεχώς αναπτύσσεται και γίνεται ολοένα και πιο ισχυρός είναι η γλώσσα προγραμματισμού Javascript. Από το 2000 και έπειτα η Javascript γίνεται ένας έμπιστος φίλος του προγραμματιστή του εμπρόσθιου τμήματος, ο οποίος αποκτά μεγάλη δύναμη χρησιμοποιώντας την στις εφαρμογές του.

Οι δυνατότητες που παρέχει η Javascrpipt είναι τόσο σημαντικές ώστε να γεννηθεί η ανάγκη ανάπτυξης πολλών πλαισίων λογισμικού (frameworks) πάνω σε αυτή από διάφορους φορείς. Ο τεράστιος αριθμός των πλαισίων λογισμικού που έχουν αναπτυχθεί, κάνει αδύνατη την μαζική ανάλυσή και μελέτη τους από έναν προγραμματιστή, ο οποίος θα περιοριστεί στη μελέτη και χρήση ενός μικρού αριθμού αυτών, ανάλογα με τις ανάγκες των εφαρμογών του.

Στην παρούσα ενότητα θα παρουσιαστούν και θα αναλυθούν τα δημοφιλέστερα, σύμφωνα με forums και διαδικτυακά άρθρα πληροφορικής, τα πιο δημοφιλή πλαίσια λογισμικού της Javascript, καθώς και συγκριτική μελέτη μεταξύ των συγκεκριμένων πλαισίων.

4.1 Τα πιο δημοφιλή σύγχρονα frameworks της javascript

Όπως προαναφέρθηκε, ο μεγάλος αριθμός των πλαισίων λογισμικού της javascript καθιστά απαγορευτική τη μαζική ανάλυσή τους, επομένως θα περιοριστούμε στα 4 πιο δημοφιλή, όπως αυτά προκύπτουν από μελέτη των τάσεων (trends) στις διαδικτυακές αναζητήσεις που αφορούν πλαίσια λογισμικού της javascript από την Google καθώς επίσης και από άρθρα πάνω στον τομέα των πλαισίων λογισμικού της Javascript.
Τα πιο δημοφιλή που προκύπτουν είναι τα εξής:

1. Backbone.js
2. AngularJS
3. Ember.js
4. React.js

4.1.1 Backbone.js

Το Backbone.js είναι ένα πλαίσιο λογισμικού μινιμαλιστικά προσανατολισμένο στο μοντέλο MVC (Model View Controller), όπως και τα περισσότερα σύγχρονα πλαίσια javascript. Πρόκειται για ένα ελαφρύ και πολύπλευρο framework που πρωτοεμφανίστηκε το 2010 από τον Jeremy Ashkenas, το μέγεθος του οποίου δεν ξεπερνούσε τα 6.3 KB.

Με μια πρώτη ματιά θα μπορούσαμε να ισχυριστούμε ότι ένα από τα κύρια χαρακτηριστικά του Backbone είναι το μικρό του μέγεθος, αλλά κάτι τέτοιο θα αδικούσε

τον πολύπλευρο χαρακτήρα του ως πλαίσιο λογισμικού. Πρακτικά χαρακτηρίζοντάς το ως πολύπλευρο, εννοούμε ότι δεν περιορίζει τον χρήστη σε ένα βασικό μηχανισμό για προτυποποίηση (templating) όπως συνήθως συμβαίνει με άλλα frameworks και έτσι δίνει στον προγραμματιστή την ελευθερία να κρίνει αυτός το πως θα κινηθεί ώστε να αναπτύξει την εφαρμογή του.

Επίσης, το γεγονός ότι είναι ελαφρύ, του δίνει πολλά συν όταν αναζητείται ένα framework για απλές εφαρμογές όπως SPAs (Single Page Apps) ή για κάποιο widget και γενικότερα, για εφαρμογές που προτεραιότητα είναι η ταχύτητα.

Το γεγονός ότι το Backbone δίνει μεγάλη ελευθερία στον χρήστη και δεν του παρέχει κάποιο μηχανισμό για προτυποποίηση, το κάνει κατά κάποιον τρόπο πιο ταιριαστό σε προγραμματιστές με μεγαλύτερη εμπειρία πάνω στην Javascript. Ένας νέος προγραμματιστής ενδέχεται να «χαθεί» μέσα στον πολύπλευρο χαρακτήρα του Backbone.

Γενικά το Backbone είναι ένα framework που κάποιος έμπειρος προγραμματιστής θα καταφέρει να αναπτύξει αποδοτικά SPAs κυρίως, αλλά για ευρύτερη χρήση γενικά στις διαδικτυακές εφαρμογές, σε περίπτωση που προτιμηθεί ως πλαίσιο λογισμικού της Javascript, θα πρέπει να υπάρξει αρκετή μελέτη και προσοχή ώστε να επιτευχθεί ένα ικανοποιητικό αποτέλεσμα.

4.1.2 Ember

Το Ember είναι ένα πλαίσιο λογισμικού που αναπτύχθηκε το 2011 από τον Yehuda Katz, μέλος των ομάδων των JQuery και Ruby on Rails. Το Ember απαιτεί τις εξαρτήσεις (dependencies) από τα πλαίσια λογισμικού JQuery και Handlebars της Javascript ώστε να έχει διαθέσιμες όλες τις λειτουργίες της.

Το Ember σαν πλαίσιο λογισμικού έχει κάποιες στάνταρ τεχνικές ώστε να χρησιμοποιηθεί σωστά. Σχεδόν κάθε βασική λειτουργία που αφορά τον τομέα των διαδικτυακών εφαρμογών έχει μια προκαθορισμένη τεχνική που καθοδηγεί τον προγραμματιστή σχετικά με το πως θα πρέπει να κινηθεί. Έτσι λοιπόν, ένας προγραμματιστής που χρησιμοποιεί Ember είναι συγκεντρωμένος σε προβλήματα που έχουν να κάνουν με την εφαρμογή που δουλεύουν αυτή καθεαυτή (δεν ανακαλύπτει ξανά τον τροχό για κάθε δουλειά του).

Βέβαια, το να απαιτείται από τον προγραμματιστή να γράψει κώδικα «Όπως θέλει η Ember» και να δίνει από ελάχιστη έως καθόλου ελευθερία, δεν είναι απαραίτητα και ότι καλύτερο. Δηλαδή υπάρχει παρόμοιος προβληματισμός με το Backbone που δίνει απόλυτη ελευθερία με λιγότερες built in λειτουργίες.

4.1.3 React

Ένα νέο σε σχέση με τα προαναφερόμενα πλαίσιο λογισμικού της Javascript, δημοσιεύτηκε το 2013 από το Facebook. Σε αντίθεση με τα 2 προηγούμενα, το React δεν είναι με τον ίδιο τρόπο στο μοντέλο client-side MVC. Σύμφωνα με το Facebook, πρόκειται για ένα πλαίσιο λογισμικού προσανατολισμένο στο View. Δεν απαιτεί καμία εξάρτηση (dependency) για να λειτουργήσει σωστά.

Ως πιο νέο πλαίσιο λογισμικού, είναι πιο γρήγορο από τα προαναφερθέντα και πιο εύκολο στην εκμάθηση. Ακόμη ένα ισχυρό χαρακτηριστικό του React είναι ότι κάθε δομικού στοιχείο (component) του αντιπροσωπεύει ένα στοιχείο της διεπαφής χρήστη (user interface) - ένα στοιχείο φόρμας / πίνακα, έναν τίτλο σελίδας κλπ.

Επιπρόσθετα, το React είναι συμβατό και με εφαρμογές κινητών τηλεφώνων. Μέχρι στιγμής υποστηρίζει εφαρμογές iOS και σύντομα θα εισαχθεί και στο android.

Από την αντίθετη πλευρά, το React δεν περιέχει templates και για την ανάπτυξη της διεπαφής χρήστη (User Interface) πρέπει συνεχώς να χρησιμοποιούνται διαφορετικά δομικά στοιχεία του λογισμικού. Έτσι, ο HTML κώδικας αναπτύσσεται μέσα στον Javascript κώδικα.

4.1.4 Anglular.js

Το AngularJS είναι ένα framework που αναπτύχθηκε από την εταιρία Brat Tech LLC και δημοσιεύτηκε το 2009, ενώ στη συνέχεια είχε και την υποστήριξη της Google η οποία ανέβασε σε σημαντικό βαθμό τη δημοτικότητά της. Η Angular δεν προσανατολίζεται αυστηρά το τυπικό MVC, εξού και ο χαρακτηρισμός της ως Model - View – Whatever framework.

Οι προκαθορισμένες λειτουργίες της Angular την κάνουν ένα αρκετά δυνατό πλαίσιο λογισμικού της Javascript που είναι και ο βασικός λόγος της μεγάλης δημοτικότητάς της.

Ένα από τα δυνατά χαρακτηριστικά της Angular είναι αυτό που ονομάζουμε αμφίδρομη αντιστοίχιση (two-way data binding). Η Angular παρέχει τη δυνατότητα της αντιστοίχισης (binding) στοιχείων (elements) του HTML κώδικα (στο κομμάτι της όψης - View), δίνοντας τη δυνατότητα στα στοιχεία αυτά τόσο να παρουσιάζουν και να επεξεργάζονται δεδομένα. Η Angular δίνει τη δυνατότητα τόσο στην Όψη (View) όσο και στο Μοντέλο (Model) να επεξεργάζονται και να ανανεώνουν δεδομένα, γι’ αυτό και η παρούσα λειτουργία ονομάζεται αμφίδρομη αντιστοίχιση.

Ένα ακόμη σημαντικό χαρακτηριστικό της Angular είναι τα πρότυπα (directives), τα οποία επεκτείνουν το λεξικό της HTML μετατρέποντας τον HTML κώδικα σε γλώσσα πιο κατανοητή και βοηθά σημαντικά την προτυποποίηση (templating). Υπάρχουν πολλά προκαθορισμένα πρότυπα στην Angular, αλλά ταυτόχρονα δίνεται και η δυνατότητα στον προγραμματιστή να αναπτύξει τα δικά του πρότυπα ώστε να μπορεί να έχει άνεση στην ανάπτυξη των δικών του προτύπων στη διεπαφή χρήστη.

Επίσης, στην Angular υπάρχει και το χαρακτηριστικό της εισαγωγής εξαρτήσεων (Dependency injection) που επιτρέπει στον προγραμματιστή να εισάγει υπηρεσίες (services) στις μεθόδους του. Πχ ας υποθέσουμε ότι κάποιος προγραμματιστής θέλει να γράψει μια συνάρτηση, η οποία θα πρέπει να χρησιμοποιήσει τη μέθοδο GET του http πρωτοκόλλου. Το μόνο που απαιτείται είναι να δώσει ένα όρισμα του http και η Angular θα φροντίσει να παρέχει ένα στιγμιότυπο (instance) αυτής της υπηρεσίας στην εμβέλεια της συνάρτησης.

Από την άλλη πλευρά, η δυνατότητα της αμφίδρομης αντιστοίχισης, όσο εντυπωσιακή κι αν ακούγεται και όσο χρήσιμη είναι, έχει προβληματίσει το χώρο της ανάπτυξης του εμπρόσθιου τμήματος των διαδικτυακών εφαρμογών σχετικά με την πολυπλοκότητα που προσδίδει στην αποσφαλμάτωση (debugging) και με την κακή επιρροή της στην απόδοση.

Γενικότερα, σε μια περίπλοκη διαδικτυακή εφαρμογή σε επίπεδο διεπαφής χρήστη θα πρέπει να υπάρξει αρκετή προσοχή στην ανάπτυξη του Angular κώδικά ώστε να μην

υπάρχει κακή απόδοση (άσκοπη χρήση υπηρεσιών, κακή χρήση της τεχνικής αμφίδρομης αντιστοίχισης κλπ).

4.2 Σύγκριση των 4 πλαισίων λογισμικού

Η Backbone, αν και ελαφρύ πλαίσιο λογισμικού, δεν θα ήταν η καλύτερη επιλογή για κάποιον προγραμματιστή χωρίς αρκετή εμπειρία λόγω της μεγάλης ελευθερίας που δίνει στην ανάπτυξη κώδικα.

Για παρόμοιο λόγο, ίσως και η Ember να μην ήταν η καλύτερη επιλογή λόγω των περιορισμών που θέτει στον προγραμματιστή να αναπτύξει κώδικα σύμφωνα με συγκεκριμένο προκατασκευασμένο τρόπο.

Γενικότερα, η βέλτιστη λύση είναι ένα framework που ναι μεν θα έχει αρκετές προεγκατεστημένες λειτουργίες, αλλά θα πρέπει να επιτρέπει και στον προγραμματιστή να τις επεκτείνει ή να φτιάξει τις δικές του. Για τον λόγο αυτό, σε ψηλή θέση βρίσκονται τα πλαίσια Angular και React.

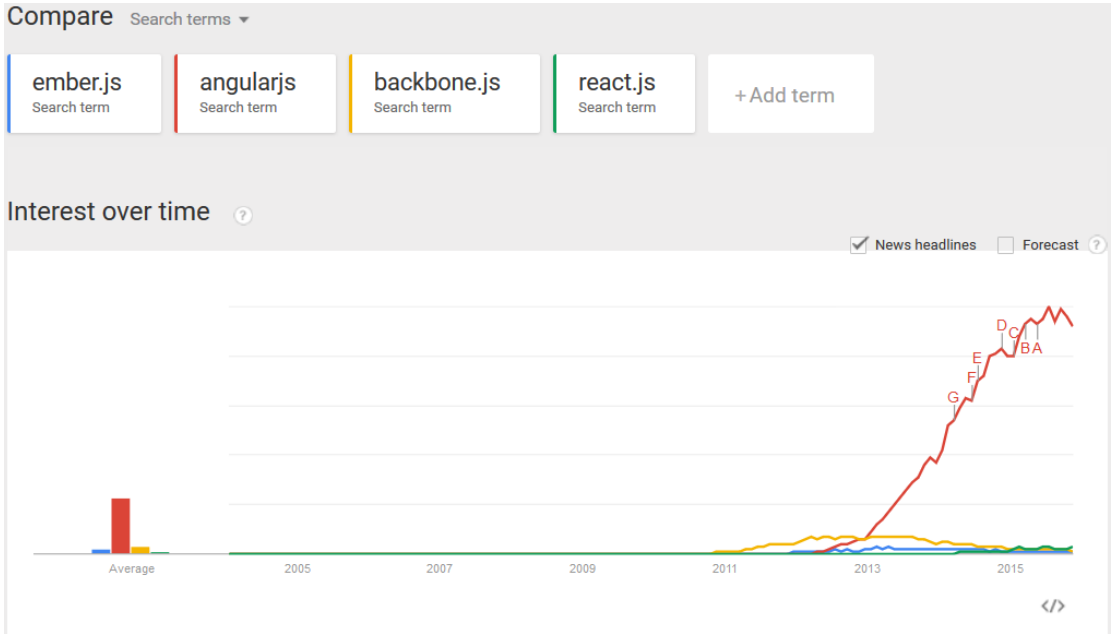
Ακόμη, θα πρέπει να ληφθεί υπ' όψιν και το κατά πόσο το πλαίσιο λογισμικού που θα χρησιμοποιηθεί θα καλύπτει, αν όχι όλες, τότε τις περισσότερες ανάγκες μιας σύγχρονης διαδικτυακής εφαρμογής.

Όπως φαίνεται στον ακόλουθο πίνακα, τόσο η Angular όσο και η Ember φαίνονται να είναι πιο «ολοκληρωμένα» πλαίσια λογισμικού, δηλαδή καλύπτουν ένα μεγαλύτερο φάσμα στον τομέα των διαδικτυακών εφαρμογών.

Πίνακας 1: Πλαίσια Λογισμικού της Javascript και ανάγκες που καλύπτουν

	UI Binding	Reusable Components	Routing
Ember.js	●	●	●
Angular.js	●	●	●
Backbone.js		●	●
React	●	●	

Επίσης σημαντικός παράγοντας είναι και η υποστήριξη του κάθε πλαισίου λογισμικού, είτε από την ομάδα ανάπτυξής του είτε από forums για προγραμματιστές. Για το λόγο αυτό, έγινε ανάλογη αναζήτηση στα trends των αναζητήσεων στο Google για καθένα από τα 4 πλαίσια λογισμικού και τα αποτελέσματα φαίνονται στην ακόλουθη εικόνα



Σχήμα 1: Ενδιαφέρον αναζήτησης των πλαισίων λογισμικού μέσω του Google Trends

Παρατηρούμε ότι από το 2012 και μετά, η Angular εκτοπίζει τα άλλα 3 πλαίσια λογισμικού στις αναζητήσεις στο google, επομένως μπορούμε να ισχυριστούμε ότι μέχρι στιγμής έχει την καλύτερη υποστήριξη στο διαδίκτυο, επομένως θα ήταν μια πολύ καλή επιλογή για την ανάπτυξη του εμπρόσθιου τμήματος μιας διαδικτυακής εφαρμογής.

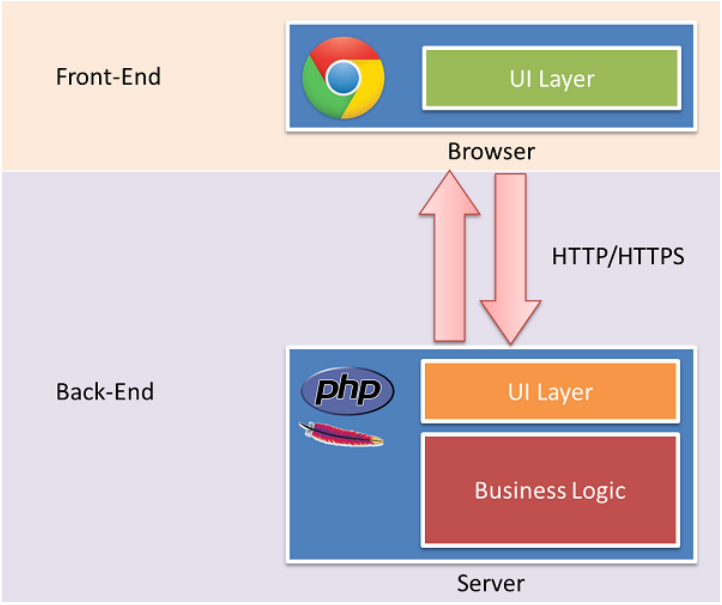
4.3 Σύστημα υποστήριξης REST API

Μία σύγχρονη αρχιτεκτονική ανάπτυξης εφαρμογών διαδικτύου για το σύστημα υποστήριξης μιας εφαρμογής είναι ο σχεδιασμός ενός Rest API που παρέχει, μέσω απλών αιτημάτων http, επικοινωνία μεταξύ του εμπρόσθιου τμήματος και των δεδομένων που είναι αποθηκευμένα στη βάση.

Με την ανάπτυξη νέων πλαισίων λογισμικού όπως της node.js, η τεχνική της ανάπτυξης ενός Rest API και ενός συστήματος υποστήριξης γενικότερα λαμβάνει νέες διαστάσεις.

4.3.1 Εισαγωγή της nodeJS στο σύστημα υποστήριξης

Τα χρόνια μεταξύ της εισαγωγής του Ajax και μέχρι προσφάτως, οι διαδικτυακές εφαρμογές είχαν την αρχιτεκτονική που φαίνεται στο παρακάτω σχήμα:



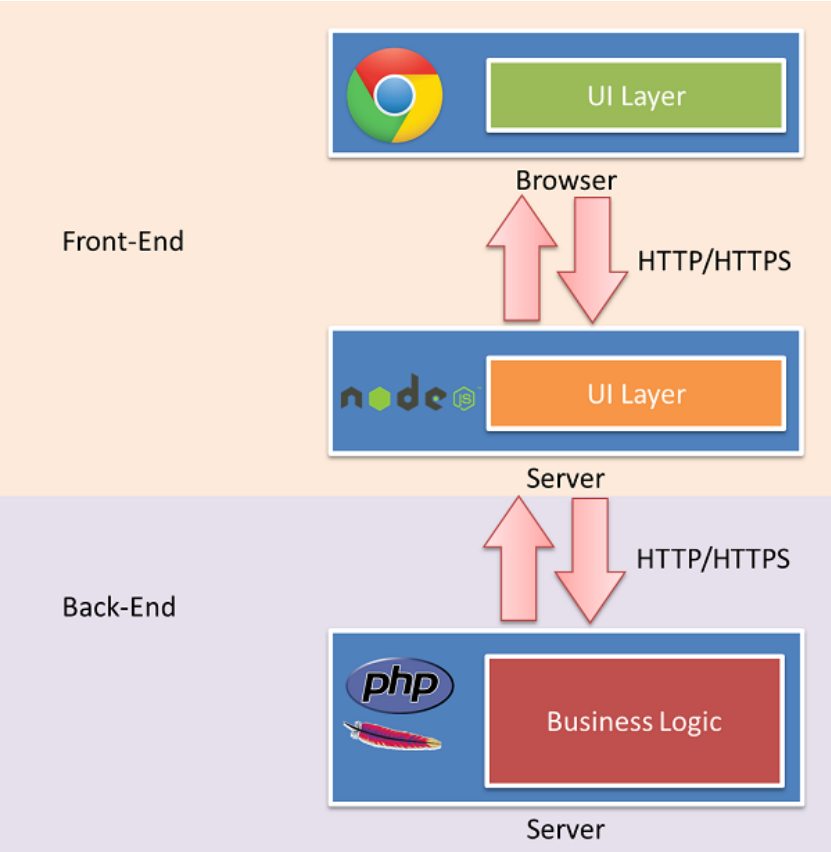
Σχήμα 2: Κλασική Αρχιτεκτονική Διαδικτυακών Εφαρμογών [18]

Όπως φαίνεται στο σχήμα, υπήρχαν ουσιαστικά 2 επίπεδα διεπαφής χρήστη, ένα στο εμπρόσθιο τμήμα και ένα στο σύστημα υποστήριξης. Το εμπρόσθιο τμήμα ουσιαστικά είχε να κάνει αποκλειστικά και μόνο με την όψη στο πρόγραμμα περιήγησης και το σύστημα υποστήριξης κυρίως με τη λογική της εφαρμογής (πχ κώδικας PHP όπως στο σχήμα που υποστηρίζει την επικοινωνία του εμπρόσθιου τμήματος με το σύστημα υποστήριξης). Βέβαια, βλέπουμε ότι το σύστημα υποστήριξης έχει ένα επίπεδο διεπαφής χρήστη που ουσιαστικά εκεί συναντιέται με το εμπρόσθιο τμήμα. Πρόκειται για ένα επίπεδο που πραγματοποιεί τη μεταβίβαση των αιτημάτων από το πρόγραμμα περιήγησης στο τμήμα της λογικής της εφαρμογής (Business Logic) και των αποκρίσεων από το τμήμα της λογικής στο πρόγραμμα περιήγησης.

Κάτι τέτοιο, όπως είναι προφανές, έπρεπε να φέρει τους προγραμματιστές του εμπρόσθιου τμήματος σε συνεννόηση με αυτούς του συστήματος υποστήριξης σχετικά με το πως θα πρέπει να χρησιμοποιηθούν τα πλαίσια λογισμικού που θα χρησιμοποιηθούν σε κάθε εφαρμογή. Επομένως, το σύστημα υποστήριξης έχριζε μεγαλύτερης προσοχής από το εμπρόσθιο τμήμα της διαδικτυακής εφαρμογής και το δεύτερο έπρεπε να φροντίζει να έρχεται σε πλήρη συνεννόηση με το πρώτο.

Με την εμφάνιση της node.js ουσιαστικά φτάσαμε στο σημείο να μπορούμε να αναπτύξουμε javascript κώδικα στον εξυπηρετητή (server). Κάτι τέτοιο θα μπορούσε να ελαφρύνει τη δουλειά του προγραμματιστή του συστήματος υποστήριξης και να δώσει τις αρμοδιότητες που αφορούν το επίπεδο διεπαφής χρήστη στον προγραμματιστή του εμπρόσθιου τμήματος και μόνο, ώστε η μοναδική αρμοδιότητα του συστήματος υποστήριξης να είναι η λογική της εφαρμογής.

Στο παρακάτω σχήμα φαίνεται η νέα αρχιτεκτονική που χρησιμοποιείται με την εισαγωγή της node.js στις σύγχρονες διαδικτυακές εφαρμογές



Σχήμα 3: Αρχιτεκτονική Διαδικτυακών Εφαρμογών μετά την είσοδο της node.js [18]

Συμπεραίνουμε λοιπόν ότι το σύστημα υποστήριξης απελευθερώνεται από το επίπεδο της διεπαφής χρήστη που τώρα πια είναι αρμοδιότητα του εμπρόσθιου τμήματος. Στην παραπάνω εικόνα παρουσιάζεται ως ξεχωριστό κομμάτι από την node.js το τμήμα της λογικής της εφαρμογής [7]. Κάτι τέτοιο δεν αποκλείει βέβαια τη δυνατότητα που παρέχει στον προγραμματιστή η node.js να μπορεί να αναπτύσσει και τη λογική της εφαρμογής σε Javascript, και θα παρουσιαστεί παρακάτω.

Γενικά, με την εισαγωγή των RESTful διεπαφών στις διαδικτυακές εφαρμογές, ολόκληρη η λογική της εφαρμογής και γενικά το σύστημα υποστήριξης μπορεί να αναπαρασταθεί με κλήσεις REST πρωτοκόλλου και η όλη λογική είναι να μπορεί μια εφαρμογή να στηθεί με τη στήριξη REST κλήσεων.

Δηλαδή, με τη χρήση της node.js και των RESTful διεπαφών, δεν απαιτείται πλέον συνεννόηση μεταξύ των προγραμματιστών του εμπρόσθιου τμήματος και του συστήματος υποστήριξης της εφαρμογής. Ο πρώτος μπορεί μέσω της node.js να κάνει REST κλήσεις στον server και ο δεύτερος να αναπτύξει σε όποια γλώσσα επιθυμεί τις κατάλληλες υπηρεσίες REST ώστε να υποστηρίζεται η εφαρμογή.

Με άλλα λόγια διαχωρίζουμε καλύτερα το εμπρόσθιο τμήμα με το σύστημα υποστήριξης. Ο εξυπηρετητής έχει ως αρμοδιότητα να φροντίζει την άρτια ροή των δεδομένων ενώ ένα πρόγραμμα πελάτη οφείλει να έχει προγραμματιστεί έτσι ώστε να γίνονται σωστά αιτήματα στον εξυπηρετητή, ανάλογα με την πλοήγηση του χρήστη.

4.3.2 Πλαίσιο λογισμικού της nodeJS – Strongloop

Ένα σύγχρονο πλαίσιο λογισμικού που εισήχθη το 2013 πάνω στη node.js είναι το Strongloop. Χτισμένο πάνω στον ανοικτό κώδικα του προϋπάρχοντος πλαισίου λογισμικού της node.js, το LoopBack, το Strongloop επιτρέπει στον προγραμματιστή να

αναπτύσει γραφικά REST APIs και να τα συνδέει με τα δεδομένα της βάσης του [8].

Το StrongLoop βοηθά στην ανάπτυξη διαδικτυακών APIs νέας γενιάς που επιχειρούνται μέσω της node.js. Επίσης περιέχει μία πληθώρα από δομές (modules) ανοικτού κώδικα της node.js και υποστηρίζει τα γνωστά πλαίσια λογισμικού της node.js, τα LoopBack και Express.

Το StrongLoop απευθύνεται σε προγραμματιστές που γράφουν node.js τόσο αρχάριους όσο και προχωρημένους. Είναι αρκετά σύγχρονο όπως και η node.js και πρόκειται για ένα ισχυρό εργαλείο στα χέρια ενός προγραμματιστή που επιθυμεί να αναπτύξει REST APIs με χρήση της node.js. Για το λόγο αυτό, θα ήταν μια άρτια επιλογή για μια διαδικτυακή εφαρμογή που στηρίζεται στο μοντέλο REST API σαν σύστημα υποστήριξης ανεπτυγμένο σε γλώσσα node.js – εμπρόσθιο τμήμα ανεπτυγμένο σε Javascript.

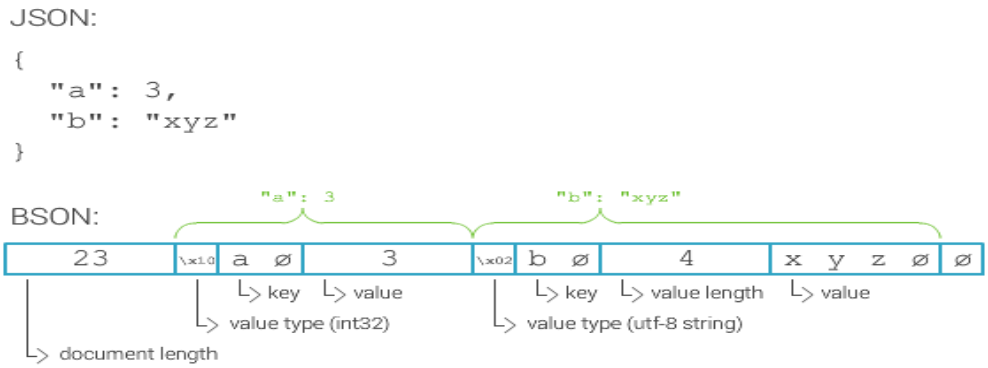
4.4 MongoDB - Μία μη Σχεσιακή Βάση δεδομένων

Η mongoDB είναι μια μη σχεσιακή βάση δεδομένων που πρωτοεμφανίστηκε το 2009. Πρόκειται για μια από τις πιο γνωστές μη σχεσιακές βάσεις δεδομένων. Ακολουθεί το πρότυπο της αρχειοστραφούς λογικής των μη σχεσιακών βάσεων δεδομένων που αναλύθηκε παραπάνω και είναι γραμμένη κυρίως σε γλώσσα C++, Javascript και C [9].

Η mongoDB παρέχει μηχανισμούς αποθήκευσης και ανάκτησης δεδομένων, καθώς και γλώσσα ώστε να γίνονται ερωτήματα (queries), σύμφωνα με το αρχειοστραφές μοντέλο. Τα δεδομένα αποθηκεύονται σε οριζόντια διαβάθμιση (horizontal scale) και έτσι παρέχεται μεγαλύτερη διαθεσιμότητα και γρηγορότερη προσπέλαση.

Επίσης είναι περιβάλλον επεξεργασίας σε πραγματικό χρόνο (real time processing) και μπορεί να αποθηκεύσει τεράστιο όγκο δεδομένων. Υποστηρίζει τεχνικές επεξεργασίας υποσυνόλου δεδομένων (clusters) και ο μέσος χρόνος επεξεργασίας (processing time) είναι της τάξης των μερικών milli-seconds.

Τα αρχεία (documents) αποθηκεύονται, όπως έχει προαναφερθεί, σε μορφή BSON η οποία μορφή επεξηγείται στην ακόλουθη εικόνα.



Εικόνα 6: JSON και BSON κωδικοποίηση [17]

Επιπλέον, αξίζει να σημειωθεί ότι η mongoDB συνεργάζεται άρτια με το πλαίσιο λογισμικού της node.js Strongloop μέσω μηχανισμών που παρέχει το ίδιο το Strongloop. Επομένως είναι μια πολύ καλή επιλογή κεντρικής βάσης δεδομένων για τις ανάγκες μιας διαδικτυακής εφαρμογής που βασίζεται στην αρχιτεκτονική REST API σε node.js και Javascript στο εμπρόσθιο τμήμα.

4.5 Πλαίσια λογισμικού της Apache για την ανάπτυξη ενός Συστήματος Προτάσεων

Στις μέρες μας και έπειτα από χρόνια μελέτης και πειραμάτων πάνω στις τεχνολογίες σχεδίασης βάσεων δεδομένων, καθώς και σε τεχνολογίες που αφορούν την ανάλυση δεδομένων – Μεγάλων Δεδομένων στην προκειμένη, είμαστε σε θέση να επιλέξουμε ανάμεσα σε διάφορες βάσεις δεδομένων και σε ακόμη περισσότερα λογισμικά που βοηθούν στην επεξεργασία (clustering, map reducing, μετρικές και αναλύσεις κλπ) Μεγάλων Δεδομένων που είναι αποθηκευμένα σε μια κεντρική βάση δεδομένων.

Στην παρούσα έρευνα, θα εστιαστούμε στην mongoDB ως κεντρική βάση δεδομένων, στο πλαίσιο λογισμικού της Apache, το Hadoop που σε συνεργασία με την mongoDB βοηθά στην επεξεργασία δεδομένων μέσω τεχνικών map reduce και στο πλαίσιο Spark, επίσης της Apache που λειτουργεί σαν ένα επιπλέον στρώμα, ανωτέρου επιπέδου από αυτό του map reduce, επεξεργασίας δεδομένων ανάμεσα στην εφαρμογή και τη βάση δεδομένων.

Με τη χρήση των δύο ανωτέρω πλαισίων λογισμικού, στόχος είναι η δημιουργία ενός Συστήματος Προτάσεων (Recommendation System). Με τη χρήση των τεχνικών map reduce του Hadoop σε συνδυασμό με τη δυνατότητα που δίνει το πλαίσιο Spark, μέσω των υψηλού επιπέδου βιβλιοθηκών που παρέχει, να αναπτύσσονται εύκολα περίπλοκες εφαρμογές ανάλυσης δεδομένων μεγάλου μεγέθους, θα μπορούσαμε να αναπτύξουμε ένα ισχυρό σύστημα προτάσεων για την εφαρμογή που περιγράφεται στην παρούσα μελέτη.

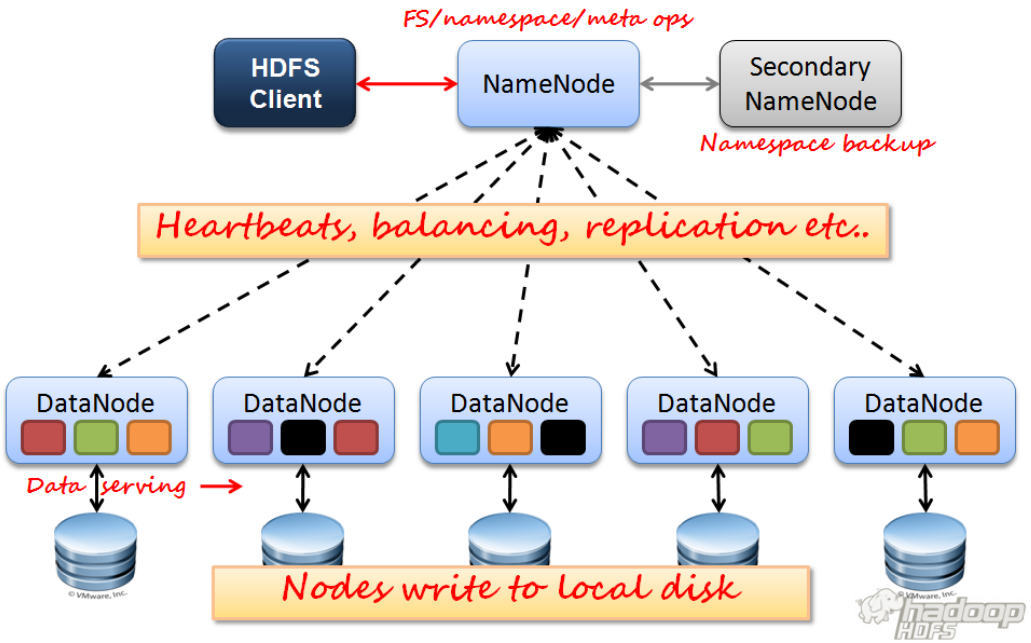
4.5.1 Hadoop

Το Hadoop είναι ένα πλαίσιο λογισμικού ανοικτού κώδικα, ανεπτυγμένου από την Apache, που κατά κύριο λόγο χρησιμοποιείται για την επεξεργασία Μεγάλων δεδομένων με έξυπνες - βέλτιστες τεχνικές. Το συγκεκριμένο πλαίσιο διανέμεται υπό την αιγίδα του Apache (Apache license) [10].

Μια βασική λειτουργία που προσφέρει το Hadoop είναι οι τεχνικές map reduce που κυρίως αξιοποιούνται όταν χρησιμοποιείται το συγκεκριμένο πλαίσιο λογισμικού. Η τεχνική αυτή χρησιμοποιείται κυρίως για τον υπολογισμό αναλυτικών δεδομένων. Στην προκειμένη περίπτωση, η τεχνική του map reduce θα χρησιμεύσει κατά κύριο λόγο ως ένας τρόπος να δοθεί μεγαλύτερη υπολογιστική ισχύς στη λογική του συστήματος προτάσεων που θα αναπτυχθεί [15].

Όπως είναι προφανές, το hadoop δεν προορίζεται για υπολογισμούς σε πραγματικό χρόνο (real time processing) από τη στιγμή που χρησιμοποιείται για υπολογισμό αναλυτικών σε Μεγάλα Δεδομένα. Ο χρόνος που απαιτείται για τον υπολογισμό σε μεγάλο όγκο δεδομένων είναι της τάξης λεπτών έως ωρών, επομένως χρησιμοποιείται σαν μια πλατφόρμα επεξεργασίας σε μη πραγματικό χρόνο.

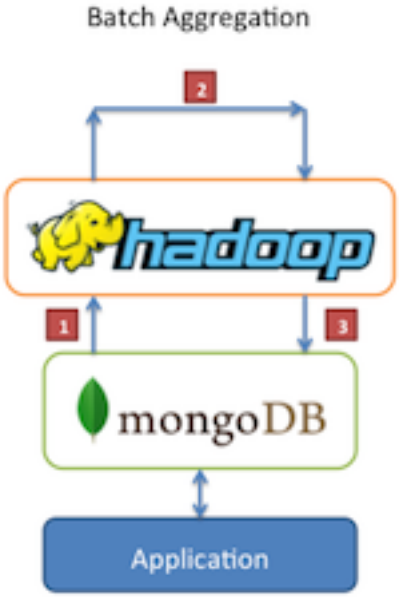
Ένα χαρακτηριστικό του Hadoop που του προσφέρει μεγάλες δυνατότητες είναι το σύστημα αρχείων του, το HDFS (HaDoop File System). Σύμφωνα με αυτό το σύστημα, κάθε διαδικασία εισόδου – εξόδου δεδομένων ή ακόμα και πιο σύνθετες λειτουργίες όπως το map reducing γίνεται με έναν ασφαλή, γρήγορο και αποδοτικό τρόπο. Η όλη διαδικασία είναι ακολουθεί την αρχή του master – slave, όπου ένας Κεντρικός Κόμβος Namenode (master) χωρίζει τη λειτουργία που καλείται να εκτελέσει σε τμήματα και τα στέλνει σε Κόμβους Επεξεργασίας Δεδομένων Datanodes (slaves), οι οποίοι είναι υπεύθυνοι να εκτελέσουν το τμήμα της εργασίας που τους αναλογεί.



Εικόνα 7: Αρχιτεκτονική του οικοσυστήματος του Hadoop [15]

Το HDFS εγγυάται ασφάλεια των δεδομένων που ανασύρθηκαν από τη βάση και προορίζονται για ανάλυση καθ' όλη τη διάρκεια οποιασδήποτε λειτουργίας και παρέχει μηχανισμούς παράκαμψης τόσο υλικού (hardware) όσο και αποτυχιών δικτύου (network failures).

Τέλος, θα πρέπει να αναφερθεί ότι το Hadoop μπορεί να συνεργαστεί πολύ καλά με τη MongoDB και βοηθά τον χρήστη να υπολογίζει αρκετά περίπλοκα αναλυτικά στοιχεία και μετρικές βάσει των δεδομένων του και να τα αποθηκεύει εκ νέου στη βάση ώστε να είναι άμεσα διαθέσιμα, όπως φαίνεται και στην παρακάτω εικόνα.

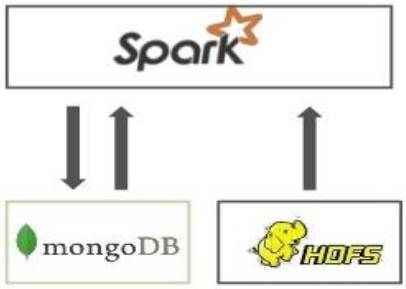


Εικόνα 8: Συνύπαρξη Hadoop και MongoDB

4.5.2 Spark

Το Spark πρόκειται επίσης για ένα πλαίσιο λογισμικού ανοικτού κώδικα που προορίζεται για αποδοτικό και γρήγορο data management. Η πρώτη του εμφάνιση έγινε το 2014 από την Apache, επομένως πρόκειται για μια πολύ νέα τεχνολογία στον τομέα της ανάλυσης δεδομένων [11].

Το Spark, όπως φαίνεται στην ακόλουθη εικόνα, θα μπορούσε να αποτελεί ένα πρόσθετο στρώμα για ένα σύστημα ανάλυσης δεδομένων σε συνεργασία με το Hadoop, έχοντας ως κεντρική βάση δεδομένων τη mongoDB.



Εικόνα 9: Συνύπαρξη MongoDB Hadoop και Spark

Το Spark παρέχει μηχανισμούς πολύ γρήγορου clustering και συνεργάζεται άρτια με το Hadoop χρησιμοποιώντας το σύστημα αρχείων του.

Οι δυνατότητες του Spark το κάνουν ένα πολύ ισχυρό όπλο σε ότι έχει να κάνει με συστημάτων προτάσεων, αφού οι σύγχρονες τεχνικές clustering που προσφέρει είναι αποδοτικές για οσοδήποτε μεγάλο όγκο δεδομένων, δομημένων ή αδόμητων.

5. ΠΑΡΟΥΣΙΑΣΗ ΤΗΣ ΠΡΟΤΥΠΗΣ ΕΦΑΡΜΟΓΗΣ

Στην παρούσα ενότητα παρουσιάζεται η διαδικτυακή εφαρμογή τύπου tripadvisor που αναπτύχθηκε με τη χρήση των τεχνολογιών που παρουσιάστηκαν παραπάνω. Γίνεται αναφορά τόσο στις απαιτήσεις - δυνατότητες της διαδικτυακής πλατφόρμας όσο και στον κώδικα με βάση τις τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξή της. Μια πιο επί της ουσίας παρουσίασή της μέσω screenshots γίνεται στο Παράρτημα 2.

Η περιγραφή ξεκινά με την παρουσίαση των στόχων της διαδικτυακής εφαρμογής, συνεχίζεται με την ανάλυση της βάσης δεδομένων που στηρίζει την εφαρμογή καθώς και με τους μηχανισμούς του συστήματος υποστήριξης (backend). Έπειτα ακολουθεί η παρουσίαση των τεχνολογιών του εμπρόσθιου τμήματος (front end) και με την ανάλυση των βασικών σελίδων της εφαρμογής και τέλος, ολοκληρώνεται με σύντομη επεξήγηση του κώδικα τόσο για το εμπρόσθιο τμήμα όσο και για το σύστημα υποστήριξης.

5.1 Στόχοι της εφαρμογής

Η διαδικτυακή εφαρμογή που αναπτύσσεται παρέχει στον χρήστη τη δυνατότητα περιήγησης σε ένα μεγάλο σύνολο ξενοδοχείων ανά τον κόσμο, βοηθώντας τον να επιλέξει με μεγαλύτερη σιγουριά το μέρος διαμονής του παρατηρώντας τις κριτικές άλλων χρηστών σε ξενοδοχεία που ενδεχομένως τον ενδιαφέρουν. Επιπλέον, δίνεται η δυνατότητα σε χρήστες που έχουν επισκεφθεί κάποιο ξενοδοχείο που βρίσκεται στη βάση δεδομένων να καταχωρήσουν την κριτική τους.

Ακόμη, δίνεται η ευκαιρία σε ιδιοκτήτες ξενοδοχείων να εγγραφούν στην εφαρμογή και να καταχωρήσουν την επιχείρησή τους ώστε να διαφημιστούν και ταυτόχρονα να έχουν μια καθαρή εικόνα για τη γνώμη των πελατών τους παρατηρώντας τις κριτικές που γίνονται από χρήστες που έχουν διαμείνει στο ξενοδοχείο τους.

Τέλος, η εφαρμογή θα παρέχει στους χρήστες μια σειρά από προτάσεις για ενδεχόμενους προορισμούς και μέρη διαμονής που θα βασίζεται πάνω στην πλοήγηση και τη δραστηριότητά τους.

5.2 Ρόλοι Χρηστών

Για τις ανάγκες της εφαρμογής, υπάρχουν 4 βασικοί ρόλοι χρηστών, οι οποίοι είναι οι εξής:

1. **Ρόλος διαχειριστή (administrator):** Κατέχει δικαιώματα προσπέλασης, εγγραφής και τροποποίησης για οποιαδήποτε είδους εγγραφή στη βάση δεδομένων. Με κατάλληλη διεπαφή χρήστη – διαχειριστή έχει πλήρη πρόσβαση σε ολόκληρη τη βάση δεδομένων.

2. **Ρόλος εγγεγραμμένου χρήστη:** Πρόκειται για χρήστες που έχουν κάνει απλή εγγραφή στην εφαρμογή. Τα δικαιώματά τους περιορίζονται στην αναζήτηση ξενοδοχείων και την ανάγνωση και προσθήκη κριτικών πάνω σε αυτά.

3. **Ρόλος επιχειρηματία (hotelOwner):** Ο χρήστες σε ρόλο επιχειρηματία έχουν τη δυνατότητα προσθήκης ενός ή και περισσότερων ξενοδοχείων στη βάση δεδομένων . Με την κατάλληλη διεπαφή, μπορούν να προσθέσουν την επιχείρησή τους στην εφαρμογή. Για να λάβει κάποιος χρήστης τον συγκεκριμένο

ρόλο, δηλώνει κατά την εγγραφή ότι είναι ιδιοκτήτης ξενοδοχείου και περιμένει έγκριση από τον διαχειριστή μέχρι να λάβει όλα τα δικαιώματα που του αναλογούν

4. **Ρόλος μη εγγεγραμμένου χρήστη:** Οι μη εγγεγραμμένοι χρήστες έχουν μόνο δυνατότητα αναζήτησης ξενοδοχείων και ανάγνωσης κριτικών πάνω σε αυτά.

5.3 Η Κεντρική Βάση Δεδομένων της εφαρμογής

Η βάση δεδομένων της εφαρμογής σχεδιάστηκε στη NoSQL βάση δεδομένων mongoDB 3.2.1

Το σχήμα της βάσης είναι τέτοιο ώστε να υποστηρίξει το σύνολο δεδομένων (dataset) «TripAdvisor Data Set» από τη σελίδα:

http://times.cs.uiuc.edu/~wang296/Data/

που παρέχεται από τη βάση δεδομένων του ιστότοπου tripadvisor.com σε μορφή JSON.

Πρόκειται για ένα σύνολο από αρχεία σε μορφή JSON με πληροφορία σχετικά με ξενοδοχεία που είναι αποθηκευμένα στη βάση δεδομένων του ιστοτόπου tripadvisor.com
Ένα αρχείο από το σύνολο δεδομένων έχει την παρακάτω μορφή:

```
{
    "HotelInfo": // Πληροφορία σχετικά με το ξενοδοχείο
    {
        "Name": "Hotel Name",
        "HotelURL": "Url",
        "Price": "$from - $to*",
        "Address": "Location of Hotel",
        "HotelID": "ID of Hotel",
        "ImgURL": "Link to hotel image"
    }

    "Reviews": // Πίνακας (array) με όλες τις κριτικές πάνω στο ξενοδοχείο
    [{ // Review #1
        "Ratings": {
            "Service": "4",
            "Cleanliness": "5",
            "Overall": "5.0",
            "Value": "4",
            "Sleep Quality": "4",
            "Rooms": "5",
            "Location": "5"
        },
        "AuthorLocation": "Location of the author",
        "Title": "Title of Review",
        "Author": "Author's username",
        "ReviewID": "ID of the Review #1",
        "Content": "Content of the Review #1"
    },
     .......,
    { // Review #N
        "Ratings": {
            "Service": "4",
            "Cleanliness": "5",
            "Overall": "5.0",
            "Value": "4",
            "Sleep Quality": "4",
            "Rooms": "5",
            "Location": "5"
        },
        "AuthorLocation": "Location of the author",
        "Title": "Title of Review",
        "Author": "Author's username",
        "ReviewID": "ID of the Review #N",
        "Content": "Content of the Review #N"
    }]
}
```

Εικόνα 10: Δομή αρχείου από το σύνολο δεδομένων

Επομένως, για τις ανάγκες του συνόλου δεδομένων το σχήμα της βάσης αναλύεται στις εξής 3 βασικές συλλογές (collections):

- **Hotel**
  Στη συλλογή Hotel αποθηκεύεται πληροφορία που αφορά το ξενοδοχείο (βλ. Παραπάνω εικόνα, πεδίο "HotelInfo").

- **Author**
  Πρόκειται για ένα υποσύνολο των απλών χρηστών που έχουν γράψει τουλάχιστον μία κριτική σε ένα ξενοδοχείο. Τα πεδία του είναι το όνομα του χρήστη που έγραψε την κριτική και η τοποθεσία του.

- **Review**
  Στη συγκεκριμένη συλλογή αποθηκεύονται κριτικές σε ξενοδοχεία. Πεδία της συλλογής Review είναι ότι φαίνεται και στην παραπάνω εικόνα με την προσθήκη ενός ακόμη πεδίου που δείχνει σε ποιο ξενοδοχείο ανήκει η κριτική. Για τις ανάγκες της εφαρμογής, η συλλογή Review θα πρέπει να περιέχει αναφορές σε αρχεία τύπου Hotel και Author ώστε να γίνεται εύκολος ο διαχωρισμός του «ποιος έκανε την κριτική» και του «σε ποιο ξενοδοχείο αντιστοιχεί αυτή η κριτική».

Για την αποθήκευση των χρηστών στη βάση, θα χρειαστεί ακόμη μια συλλογή User καθώς και συλλογή που θα αποθηκεύονται οι ρόλοι χρηστών. Πιο αναλυτικά παρακάτω.

Για το γέμισμα της βάσης με τα δεδομένα που παρέχονται από την ιστοσελίδα tripadvisor.com, αναπτύχθηκε σύντομο πρόγραμμα σε γλώσσα προγραμματισμού Java με τη χρήση των πακέτων mongo.jar για τη σύνδεση με τη βάση δεδομένων mongoDB και του json-simple-1.1.jar για τη διαχείριση αντικειμένων JSON που προκύπτουν από τα αρχεία του συνόλου δεδομένων.

5.4 Σύστημα υποστήριξης της εφαρμογής (backend)

Το σύστημα υποστήριξης ακολουθεί τις αρχές της υπηρεσιοστραφούς αρχιτεκτονικής (SOA) και πρόκειται για ένα REST Api που είναι γραμμένο σε γλώσσα Javascript – NodeJS με χρήση του πλαισίου λογισμικού Strongloop.

Με τη χρήση του Strongloop γίνεται εύκολα η σύνδεση με τη βάση δεδομένων και με την αυτόματη παραγωγή κώδικα nodeJS από το ίδιο το πλαίσιο λογισμικού, εγκαθίστανται γρήγορα τα βασικά endpoints του REST Api.

Επιπρόσθετα, το Strongloop παρέχει πλαίσιο για την ανάπτυξη συστήματος που αφορά τους χρήστες της εφαρμογής. Υπάρχουν υλοποιήσεις κάποιων βασικών ρόλων χρηστών, όπως για παράδειγμα του «εγγεγραμμένου χρήστη» (authenticated), αλλά επιτρέπει και τη δημιουργία στατικών ρόλων ανάλογα με τις απαιτήσεις της εφαρμογής.

Με τη χρήση λοιπόν του πλαισίου για το σύστημα διαχείρισης χρηστών που παρέχει το Strongloop δημιουργήθηκαν οι βασικοί ρόλοι που απαιτεί η εφαρμογή και δόθηκαν τα ανάλογα δικαιώματα πάνω στις εγγραφές στη βάση δεδομένων σε κάθε ρόλο χρήστη.

Για τον καθορισμό των μοντέλων του REST Api, δηλαδή μιας περιγραφής των όψεων της βάσης, χρησιμοποιήθηκε το εργαλείο arc του Strongloop που παρέχει ευκολία στον καθορισμό των μοντέλων με γραφικό περιβάλλον που προσφέρει συγκεκριμένη διεπαφή χρήστη.

Όσον αφορά τη σύνδεση του συστήματος υποστήριξης με τη βάση δεδομένων, χρησιμοποιήθηκε η υπηρεσία «loopback-connector-mongodb» που φροντίζει για τις αλληλεπιδράσεις του εμπρόσθιου τμήματος της εφαρμογής με τη βάση.

5.5 Το Εμπρόσθιο τμήμα (front end) της εφαρμογής

Η διεπαφή χρήστη χωρίζεται στις εξής βασικές σελίδες:

- **Αρχική σελίδα**: Πρόκειται για την σελίδα καλωσορίσματος που εμφανίζεται κατά τη φόρτωση της διαδικτυακής εφαρμογής από κάποιον φυλλομετρητή.

- **Σελίδα διαχειριστή**: Πρακτικά είναι η διεπαφή που δίνει στον διαχειριστή της σελίδας πλήρη πρόσβαση και όλα τα δικαιώματα σε κάθε εγγραφή της βάσης.

- **Σελίδα αποτελεσμάτων αναζήτησης ξενοδοχείων:** Στη σελίδα αυτή παρουσιάζονται τα αποτελέσματα της αναζήτησης μέσω της βασικής φόρμας αναζήτησης ξενοδοχείων της εφαρμογής.

- **Σελίδα εγγραφής νέου χρήστη:** Πρόκειται για τη σελίδα όπου ένας μη εγγεγραμμένος χρήστης κάνει εγγραφή στην εφαρμογή

- **Σελίδα ξενοδοχείου:** Στη συγκεκριμένη σελίδα παρουσιάζονται οι λεπτομέρειες ενός ξενοδοχείου μαζί με τις κριτικές που έχουν γραφεί για το ξενοδοχείο αυτό.

- **Σελίδα χρήστη:** Κάθε χρήστης έχει την προσωπική του σελίδα. Οι απλοί χρήστες έχουν πλήρη πρόσβαση στην δραστηριότητά τους (πρακτικά στις κριτικές που έχουν γράψει), ενώ οι χρήστες επιχειρηματίες έχουν πρόσβαση στις εταιρίες που έχουν αποθηκεύσει στη βάση και δυνατότητα αποθήκευσης περισσοτέρων ή επεξεργασίας των ήδη υπαρχόντων.

Για την ανάπτυξη του εμπρόσθιου τμήματος, χρησιμοποιήθηκε γλώσσα προγραμματισμού Javascript με τη βοήθεια του πλαισίου λογισμικού Angular καθώς επίσης και html5 με CSS Bootstrap για τον σχεδιασμό των βασικών σελίδων της διεπαφής.

Όσον αφορά τις τεχνικές που ακολουθήθηκαν ως προς την επικοινωνία εμπρόσθιου τμήματος με το σύστημα υποστήριξης, τον ρόλο του ενδιάμεσου τον καλύπτει η AngularJS μέσω διαδικτυακών υπηρεσιών που αναπτύχθηκαν ώστε να δοθεί η δυνατότητα επικοινωνίας του χρήστη με το REST Api με σκοπό την αλληλεπίδραση με τη βάση δεδομένων. Για κάθε αλληλεπίδραση με τη βάση δεδομένων, όπου κάτι τέτοιο είναι απαραίτητο, υπάρχει μια διαδικτυακή υπηρεσία που επικοινωνεί με το REST Api της εφαρμογής.

Όσον αφορά την οργάνωση των διαφορετικών σελίδων της εφαρμογής, αξιοποιήθηκε η τεχνική routing που παρέχει η AngularJS και μάλιστα η βιβλιοθήκη ui-routing η οποία προτιμήθηκε έναντι της ng-routing λόγω των μεγαλύτερων δυνατοτήτων που προσφέρει σε θέματα καλύτερης οργάνωσης των όψεων της διεπαφής χρήστη.

Για τη σελιδοποίηση των αποτελεσμάτων αναζήτησης (και γενικότερα της απεικόνισης μεγάλων λιστών με τρόπο τέτοιο ώστε να αποφεύγεται το scroll down) χρησιμοποιήθηκε η βιβλιοθήκη της AngularJS “angularUtils-pagination”

5.6 Κώδικας της εφαρμογής

Στην παρούσα ενότητα γίνεται παρουσίαση - επεξήγηση του κώδικα που αναπτύχθηκε τόσο για το εμπρόσθιο τμήμα της εφαρμογής όσο και για το σύστημα υποστήριξης. Αναλύονται οι τεχνολογίες που χρησιμοποιήθηκαν και οι απαραίτητες ρυθμίσεις που απαιτούνται για την ομαλή ροή των δεδομένων και την ορθή λειτουργία της διαδικτυακής εφαρμογής.

5.6.1 Σύντομη περιγραφή των αρχείων της εφαρμογής

Ο φάκελος της εφαρμογής (tripadvisor) περιέχει τους εξής υποφακέλους:

- client
  Πρόκειται για τον φάκελο που αποθηκεύονται τα αρχεία με τον κώδικα που αφορά το εμπρόσθιο τμήμα της εφαρμογής. Για τη συγκεκριμένη εφαρμογή, ο φάκελος client περιλαμβάνει αρχεία με κώδικα html, css και javascript (AngularJS).

- common
  Μέσα στον φάκελο common περιέχονται αρχεία json και javascript που καθορίζουν τα μοντέλα της εφαρμογής, δηλαδή κάτι σαν απεικόνιση της βάσης δεδομένων.

- node_modules
  Στον συγκεκριμένο φάκελο περιέχονται όλες οι βιβλιοθήκες που υποστηρίζουν τη λειτουργία του συστήματος υποστήριξης. Πρόκειται κυρίως για κώδικα που παράγει από μόνο του το πλαίσιο λογισμικού strongloop κατά την αρχικοποίηση της εφαρμογής.

- server
  Στον φάκελο server περιέχονται απλά json και js αρχεία που ρυθμίζουν τη συμπεριφορά του συστήματος υποστήριξης.

5.6.2 Κώδικας εμπρόσθιου τμήματος

Ό,τι αφορά το εμπρόσθιο τμήμα της εφαρμογής περιέχεται στον φάκελο client που περιγράφεται παραπάνω.

Το βασικό αρχείο που περιέχει ο φάκελος είναι το index.html που ουσιαστικά αφορά την αρχική και βασική σελίδα της εφαρμογής.

Επίσης, υπάρχει και μια σειρά από υποφάκελους Ας δούμε πιο αναλυτικά τους υποφάκελους με τα περιεχόμενά τους:

Φάκελος CSS

Πρόκειται για τον φάκελο που φιλοξενεί τα αρχεία με κώδικα CSS. Αν και γενικά για τις ανάγκες της συγκεκριμένης εφαρμογής, ό,τι έχει να κάνει με το σχεδιαστικό τμήμα (δηλαδή με τον κώδικα CSS) καλύπτεται κατά κύριο λόγο από το πλαίσιο λογισμικού Bootstrap, χρειάστηκε και κάποιος σύντομος κώδικας CSS.

Φάκελος images

Περιέχει τις εικόνες που χρησιμοποιούνται στην εφαρμογή. Ουσιαστικά πρόκειται για στατικές εικόνες που ενδεχομένως να υπάρχουν στην εφαρμογή, όπως για παράδειγμα το logo της ιστοσελίδας

Φάκελος js

Περιέχει όλα τα αρχεία javascript – AngularJS που στηρίζουν το εμπρόσθιο τμήμα.

Το βασικό αρχείο που περιέχεται στον φάκελο είναι το αρχείο app.js που ουσιαστικά ορίζει την Angular εφαρμογή που στηρίζει το εμπρόσθιο τμήμα. Στο συγκεκριμένο αρχείο γίνονται οι βασικές ρυθμίσεις που αφορούν την οργάνωση των όψεων (views) της εφαρμογής. Με άλλα λόγια, στο αρχείο app.js αναπτύσσεται ο κώδικας που χρησιμοποιεί την τεχνική routing της AngularJS.

Περιέχονται ακόμη δύο υποφάκελοι που περιέχουν αρχεία γραμμένα σε AngularJS

- Υποφάκελος controllers
  Περιέχει τους controllers για την κάθε όψη - σελίδα (view) της εφαρμογής. Τα αρχεία των controllers είναι κατάλληλα δομημένα ώστε να γίνεται ξεκάθαρο το ποια σελίδα στηρίζει ο κάθε controller.

- Υποφάκελος services
  Περιέχει δύο αρχεία που ουσιαστικά είναι διαδικτυακές υπηρεσίες οι οποίες δίνουν τη δυνατότητα επικοινωνίας του εμπρόσθιου τμήματος με το REST Api.
  Το αρχείο lb-services.js είναι η βασική διεπαφή που μέσω αυτής μπορεί η εφαρμογή να επικοινωνήσει με το REST Api και κατ' επέκταση με τη βάση δεδομένων.
  Το αρχείο auth-service.js είναι μια διεπαφή που επίσης επικοινωνεί με τη βάση, χρησιμοποιώντας πάντα το βασικό αρχείο – διεπαφή lb-services.js ώστε να αλληλεπιδρά με τη συλλογή των χρηστών της βάσης.
  Η βασική ιδέα είναι να υπάρχει αυτό το επίπεδο μεταξύ της εφαρμογής και της διεπαφής lb-services.js ώστε να μπορεί να γίνεται σωστά και πιο εύκολα και καθαρά η πιστοποίηση χρήστη.

Φάκελος vendor

Εδώ περιέχονται όλα τα αρχεία πηγαίου κώδικα των πλαισίων λογισμικού που χρησιμοποιούνται. Ουσιαστικά πρόκειται για αρχεία κώδικα που παρέχουν οι ομάδες ανάπτυξης των πλαισίων λογισμικού.

Φάκελος views

Στον φάκελο αυτό υπάρχουν αρχεία με τον html κώδικα κάθε όψης – σελίδας της εφαρμογής. Ο σχεδιασμός της δομής των σελίδων της εφαρμογής, δηλαδή το πως συνδέονται και κάτω από ποιες συνθήκες εμφανίζεται η καθεμία έχει γίνει στο αρχείο app.js που αναφέρεται παραπάνω (στο σημείο που αναφέρεται ο φάκελος js).

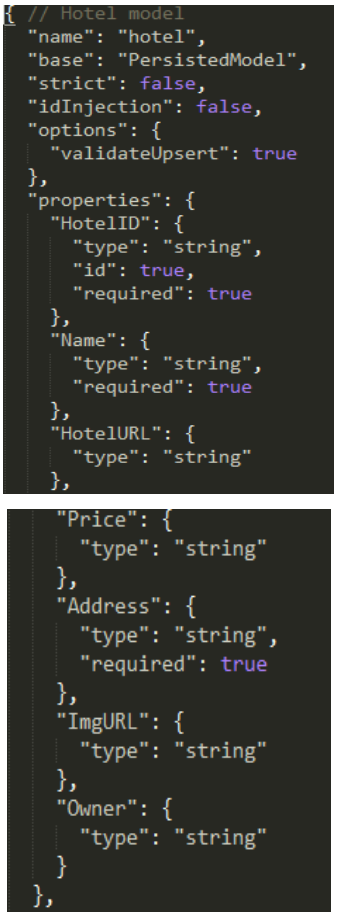
5.6.3 Κώδικας συστήματος υποστήριξης

Ο κώδικας που αφορά το σύστημα υποστήριξης περιλαμβάνεται στους υπόλοιπους φακέλους. Ο πρώτος φάκελος είναι ο common:

Ο φάκελος common περιέχει έναν υποφάκελο με το όνομα models μέσα στον οποίο περιέχονται τα μοντέλα της εφαρμογής. Πρακτικά πρόκειται για αρχεία json που αναπαριστούν τις συλλογές που αποθηκεύονται στη βάση.

Για παράδειγμα, ας δούμε ένα από τα json αρχεία που είναι αποθηκευμένα στον φάκελο models. Αναλύουμε το αρχείο hotel.json:

Όπως δηλώνει και το όνομα, πρόκειται για την αναπαράσταση της συλλογής των ξενοδοχείων που είναι αποθηκευμένα στη βάση.



```
{ // Hotel model
  "name": "hotel",
  "base": "PersistedModel",
  "strict": false,
  "idInjection": false,
  "options": {
    "validateUpsert": true
  },
  "properties": {
    "HotelID": {
      "type": "string",
      "id": true,
      "required": true
    },
    "Name": {
      "type": "string",
      "required": true
    },
    "HotelURL": {
      "type": "string"
    },
    "Price": {
      "type": "string"
    },
    "Address": {
      "type": "string",
      "required": true
    },
    "ImgURL": {
      "type": "string"
    },
    "Owner": {
      "type": "string"
    }
  },
```

Εικόνα 11: Παράδειγμα αναπαράστασης μοντέλου με τη χρήση του Strongloop

Όπως φαίνεται και στην εικόνα, μέσα στο json αρχείο, ορίζεται ο τρόπος αναπαράστασης της εγγραφής ενός ξενοδοχείου στη βάση. Σε πρώτο στάδιο, όπως βλέπουμε, ορίζουμε τα πεδία της εγγραφής και καθορίζουμε και το πρωτεύον κλειδί που στην περίπτωσή μας είναι το HotelID. Πρακτικά, αυτό σημαίνει ότι κάθε φορά που επιχειρείται η αποθήκευση μιας εγγραφής ξενοδοχείου στη βάση, το πεδίο HotelID θα είναι το πρωτεύον κλειδί της εγγραφής (Μηχανισμός που προσφέρει το πλαίσιο λογισμικού Strongloop).

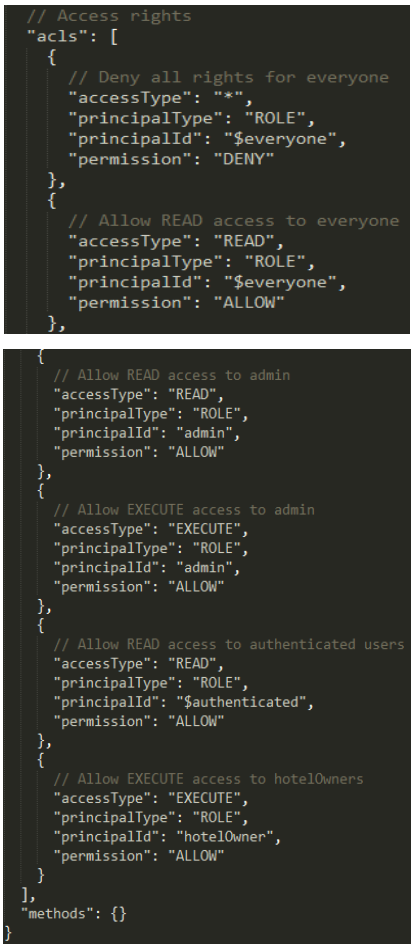
Στη συνέχεια καθορίζουμε τη σχέση που μπορεί να έχει αυτό το μοντέλο με άλλα μοντέλα της εφαρμογής μας. Στην περίπτωσή μας έχουμε την εξής σχέση:

Εικόνα 12: Καθορισμός σχέσεων μοντέλου με άλλο μοντέλο

Ουσιαστικά, σε αυτό το σημείο καθορίζουμε ότι το μοντέλο Hotel έχει μια σχέση τύπου “Has Many” με το μοντέλο Review και σαν ξένο κλειδί για τον καθορισμό της σχέσης αυτής χρησιμοποιούμε το πεδίο του μοντέλου Review που λέγεται “Hotel”. Με αυτό τον τρόπο ορίζουμε ότι ένα ξενοδοχείο μπορεί να έχει πολλές κριτικές.

Ο καθορισμός σχέσεων δεν έχει να κάνει με το σχήμα της βάσης, αφορά το REST Api και μόνο. Δηλαδή, για τη βάση δεδομένων, που δεν είναι σχεσιακή, ο όρος «ξένο κλειδί» δε σημαίνει απολύτως τίποτα, όμως μια σχέση μεταξύ των μοντέλων του REST Api βοηθά και στην κατανόηση του γενικού μοντέλου, αλλά και στον καλύτερο σχεδιασμό των endpoints του API.

Τέλος, ολοκληρώνουμε τον ορισμό ενός μοντέλου δίνοντας δικαιώματα πρόσβασης σε όποια ομάδα χρηστών κρίνουμε απαραίτητο ή αντίστοιχα εμποδίζουμε ομάδες χρηστών να έχουν πρόσβαση. Πιο αναλυτικά:



Εικόνα 13: Δικαιώματα που κατέχει κάθε ρόλος χρήστη πάνω στο μοντέλο

Σε αυτό το σημείο, δίνονται δικαιώματα πρόσβασης σε ομάδες - ρόλους χρηστών που έχουμε καθορίσει.

Οι ρόλοι $everyone και $authenticated είναι ρόλοι που καθορίζει από μόνο του το πλαίσιο λογισμικού Strongloop και ουσιαστικά ξεχωρίζουν τους χρήστες σε εγγεγραμμένους και μη.

Παρατηρούμε λοιπόν από την εικόνα ότι για το συγκεκριμένο μοντέλο, οι εγγεγραμμένοι και μη χρήστες έχουν μόνο δικαιώματα ανάγνωσης πάνω στο μοντέλο Hotel, ενώ δικαιώματα εγγραφής και τροποποίησης έχουν μόνο ο ρόλος του διαχειριστή και του επιχειρηματία (hotelOwner).

5.6.4 Βιβλιοθήκες του πλαισίου λογισμικού Strongloop

Ο επόμενος φάκελος που αφορά σύστημα υποστήριξης είναι ο φάκελος node_modules. Στον συγκεκριμένο φάκελο περιέχονται όλες οι βιβλιοθήκες που απατούνται από το πλαίσιο λογισμικού ώστε να δημιουργήσει αυτόματα τα βασικά endpoints του REST Api.

Οι βιβλιοθήκες είναι γραμμένες σε Javascript (σε nodeJS για την ακρίβεια), μαζί με κάποια αρχεία json που όπως έχουμε δει ως τώρα χρησιμοποιούνται όποτε χρειάζονται αρχεία ρυθμίσεων.

Στον συγκεκριμένο φάκελο, υπάρχει ο υποφάκελος loopback που περιέχει τα βασικά μοντέλα που δημιουργεί το Strongloop κατά την αρχικοποίηση μιας εφαρμογής.

Τα βασικά αυτά μοντέλα δεν είναι άλλα από αυτά που βοηθούν στο σύστημα ταυτοποίησης χρηστών. Βρίσκονται όλα στον υποφάκελο *loopback/common/models*. Αυτά που χρησιμοποιούνται στην παρούσα εφαρμογή είναι τα εξής:

- user.json
  Ουσιαστικά πρόκειται για τους χρήστες που κάνουν εγγραφή στην εφαρμογή μέσω της σελίδας register.html

- role.json
  Το συγκεκριμένο μοντέλο αναπαριστά τους διαφορετικούς στατικούς ρόλους που έχουν καθοριστεί από τον προγραμματιστή στην εφαρμογή, στην περίπτωσή μας τους ρόλους admin και hotelOwner.

- access-token.json
  Είναι ένα είδος ταυτότητας πρόσβασης σύμφωνα με την οποία, όταν ένας εγγεγραμμένος χρήστης συνδέεται με το λογαριασμό του στην εφαρμογή, γίνεται η ταυτοποίησή του λογαριασμού και του ρόλου του.

- role-mapping.json
  Ουσιαστικά πρόκειται για την σύνδεση των χρηστών με έναν ή περισσότερους ρόλους.

Όλα τα προαναφερθέντα μοντέλα αποθηκεύονται στη βάση δεδομένων με ρύθμιση η οποία θα παρουσιαστεί παρακάτω.

Επίσης, έχει δοθεί πλήρης πρόσβαση σε αυτά τα μοντέλα στον διαχειριστή της εφαρμογής και μόνο σε αυτόν.

Αξίζει εδώ να σημειωθεί ότι το Strongloop με τους μηχανισμούς που προσφέρει φροντίζει ώστε κατά την εγγραφή ενός χρήστη, ο κωδικός που δηλώνεται να κωδικοποιείται κατά την αποθήκευση στη βάση έτσι ώστε να υπάρχει ασφάλεια και να μην έχει κανείς πρόσβαση σε αυτό τον προσωπικό κωδικό.

Επίσης, με κανέναν τρόπο δεν μπορεί μέσω του REST Api να σταλεί ο κωδικός ενός χρήστη ακόμη κι αν ο ίδιος ο διαχειριστής της εφαρμογής αναζητήσει κάποια εγγραφή χρήστη.

Τέλος, θα πρέπει να αναφερθεί η βασική βιβλιοθήκη που χρησιμοποιείται για τη σύνδεση του Strongloop με τη βάση δεδομένων mongoDB που είναι αποθηκευμένη επίσης στον φάκελο common_modules.

Η βιβλιοθήκη ονομάζεται loopback-connector-mongodb και ο κώδικας για τη σύνδεση αυτή βρίσκεται στον ομώνυμο φάκελο.

5.6.5 Ρυθμίσεις του εξυπηρετητή

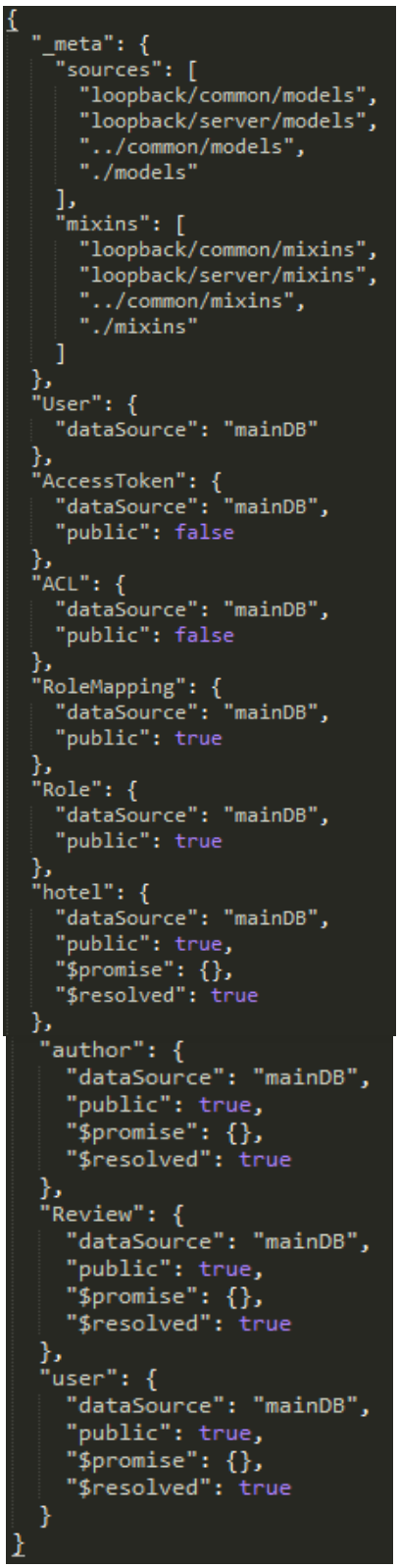
Οι ρυθμίσεις που αφορούν τον εξυπηρετητή βρίσκονται στον φάκελο server.

Με την τροποποίηση των json αρχείων του φακέλου, μπορούν να ρυθμιστούν διάφοροι παράμετροι της εφαρμογής, όπως για παράδειγμα το ποια είναι η βάση με την οποία θα επικοινωνεί το REST Api όπως φαίνεται στο αρχείο datasources.json.

```
{
  "db": {
    "name": "db",
    "connector": "memory"
  },
  "mainDB": {
    "host": "localhost",
    "database": "tripadvisorDB",
    "name": "mainDB",
    "connector": "mongodb"
  }
}
```

Εικόνα 14: Ρυθμίσεις εξυπηρετητή για επικοινωνία με τη βάση δεδομένων

Στον ίδιο φάκελο, γίνεται και ο καθορισμός της μεθόδου αποθήκευσης των μοντέλων που έχουν δημιουργηθεί, δηλαδή το που θα αποθηκεύεται κάθε μοντέλο (αρχείο model-config.json).



```
{
  "_meta": {
    "sources": [
      "loopback/common/models",
      "loopback/server/models",
      "../common/models",
      "./models"
    ],
    "mixins": [
      "loopback/common/mixins",
      "loopback/server/mixins",
      "../common/mixins",
      "./mixins"
    ]
  },
  "User": {
    "dataSource": "mainDB"
  },
  "AccessToken": {
    "dataSource": "mainDB",
    "public": false
  },
  "ACL": {
    "dataSource": "mainDB",
    "public": false
  },
  "RoleMapping": {
    "dataSource": "mainDB",
    "public": true
  },
  "Role": {
    "dataSource": "mainDB",
    "public": true
  },
  "hotel": {
    "dataSource": "mainDB",
    "public": true,
    "$promise": {},
    "$resolved": true
  },
  "author": {
    "dataSource": "mainDB",
    "public": true,
    "$promise": {},
    "$resolved": true
  },
  "Review": {
    "dataSource": "mainDB",
    "public": true,
    "$promise": {},
    "$resolved": true
  },
  "user": {
    "dataSource": "mainDB",
    "public": true,
    "$promise": {},
    "$resolved": true
  }
}
```


Εικόνα 15: Ρυθμίσεις εξυπηρετητή - Καθορισμός μοντέλων και ασφάλειας

Παρατηρούμε λοιπόν ότι κάθε μας μοντέλο το θέλουμε αποθηκευμένο στη βάση mainDB που έχει καθοριστεί παραπάνω ότι πρόκειται για την κεντρική βάση δεδομένων της εφαρμογής (αρχείο datasources.json).

Όλα τα υπόλοιπα json αρχεία που βρίσκονται στον φάκελο της εφαρμογής εκτός των φακέλων που έχουν αναπτυχθεί ως τώρα, είναι αρχεία που παράγει αυτόματα το

Strongloop και επιτρέπουν μικρές αλλαγές και ρυθμίσεις που αφορούν τον εξυπηρετητή – node server.

6. ΠΑΡΟΥΣΙΑΣΗ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ ΠΡΟΤΑΣΕΩΝ

Το σύστημα προτάσεων για την εφαρμογή τύπου tripadvisor αναπτύχθηκε με χρήση του λογισμικού Hadoop και Spark της Apache. Όσον αφορά την ανάπτυξη κώδικα, χρησιμοποιήθηκε ως επί το πλείστον η γλώσσα προγραμματισμού Java και Scala.

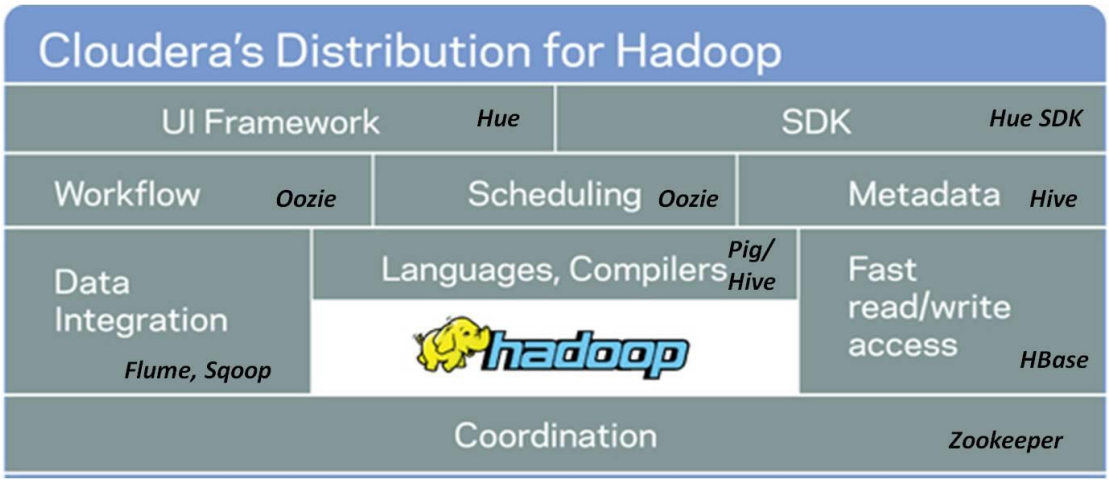
Για την υποστήριξη του συστήματος προτάσεων προτιμήθηκε περιβάλλον Linux, και για την ακρίβεια CentOS με τη χρήση εικονικού μηχανήματος (Virtual Machine) από το λογισμικό Virtual Box της Oracle.

Όσον αφορά το τμήμα των προτάσεων, χρησιμοποιήθηκε το σύστημα αρχείων (Hadoop File System - HDFS) του λογισμικού Hadoop σε συνεργασία με τις βιβλιοθήκες του Spark. Επιγραμματικά, το σύστημα αρχείων του Hadoop επικοινωνεί με τη βάση δεδομένων – mongoDB μέσω διεργασιών Map Reduce γραμμένες σε γλώσσα Java ώστε να κρατήσει πληροφορία που θα επεξεργαστεί το Spark γραμμένο σε Scala. Για την εξαγωγή αποτελεσμάτων, χρησιμοποιήθηκαν οι βιβλιοθήκες mllib του Spark που με χρήση του αλγορίθμου collaborative filtering, εξάγει προτάσεις αντικειμένων (ξενοδοχείων στην προκειμένη). Τέλος, τα αποτελέσματα αποστέλλονται στη βάση δεδομένων από το ίδιο το Spark. Με τον συγκεκριμένο τρόπο, επιτυγχάνεται η επεξεργασία πληροφορίας από τη βάση δεδομένων ανεξάρτητα από το online τμήμα της εφαρμογής και εξασφαλίζεται η ασφάλεια των δεδομένων και της σταθερότητας της εφαρμογής.

6.1 Σύστημα CentOS και χρήση Hadoop Single Node Cluster

Για την καλύτερη εμπειρία χρήσης του λογισμικού Hadoop και την καλύτερη εξομοίωση της αρχιτεκτονικής του, προτιμήθηκε το λειτουργικό σύστημα Linux CentOS με χρήση εικονικού μηχανήματος.

Η Cloudera προσφέρει εικόνες (images) για εικονικά μηχανήματα με λογισμικό CentOS που έχουν προεγκατεστημένο το Hadoop σε Single Node Cluster, δηλαδή σύστημα Hadoop με έναν κύριο κόμβο (Master Namenode) και έναν κόμβο εργάτη (Worker Node). Επίσης προσφέρει εργαλεία για την καλύτερη εποπτεία της εκτέλεσης διεργασιών map reduce του Hadoop, καθώς και του συστήματος αρχείων του Hadoop (HDFS), όπως το εργαλείο Hue .



Εικόνα 16: Διανομή λειτουργικού CentOS με προεγκατεστημένο το Hadoop και εργαλεία [21]

Για τη συγκεκριμένη εφαρμογή, χρησιμοποιήθηκε η εικόνα cdh 5.7.0 της cloudera για το λογισμικό VirtualBox της Oracle.

6.2 Γλώσσες προγραμματισμού και τεχνολογίες που χρησιμοποιήθηκαν

Όσον αφορά τις γλώσσες προγραμματισμού που χρησιμοποιούνται για τα πλαίσια λογισμικού που χρησιμοποιήθηκαν, υπάρχει μια πληθώρα από γλώσσες και τεχνολογίες που προσφέρονται για το συγκεκριμένο σκοπό.

6.2.1 Γλώσσα προγραμματισμού για το οικοσύστημα του Hadoop

Το πλαίσιο λογισμικού Apache Hadoop, πρόκειται για λογισμικό γραμμένο σε γλώσσα Java. Υπάρχουν διάφορες τεχνολογίες που θα έκαναν τη συγγραφή μιας διεργασίας map reduce του Hadoop πολύ πιο εύκολη, όπως για παράδειγμα το Pig που είναι πλαίσιο λογισμικού αρκετά υψηλού επιπέδου της Apache για εύκολη συγγραφή διεργασιών map reduce. Αν και η ύπαρξη τέτοιου είδους τεχνολογιών μπορεί να κάνει τη δουλειά ενός προγραμματιστή αρκετά ευκολότερη σε ό,τι αφορά τη συγγραφή απλών προγραμμάτων map reduce, θα ήταν προτιμότερο να επιλεχθεί η Java ως γλώσσα προγραμματισμού για το Hadoop.

Στο αναπτυχθέν Σύστημα Προτάσεων, η χρήση του Hadoop περιορίζεται στην εκμετάλλευση του συστήματος αρχείων του, του HDFS, ενώ η ανάλυση των δεδομένων είναι αρμοδιότητα του Apache Spark. Επομένως δεν απαιτούνται πολύπλοκες διεργασίες, παρά μόνον 1 πρόγραμμα τύπου map reduce που ουσιαστικά υποστηρίζει την επικοινωνία της βάσης δεδομένων με το σύστημα αρχείων του Hadoop.

Για το λόγο αυτό λοιπόν, η γλώσσα προγραμματισμού που θα χρησιμοποιηθεί για το πλαίσιο λογισμικού Hadoop είναι η Java με όλες τις απαραίτητες βιβλιοθήκες που κάνουν εφικτή τη λειτουργία του Hadoop.

6.2.2 Γλώσσες προγραμματισμού για το Spark

Το πλαίσιο λογισμικού Apache Spark είναι το λογισμικό που καθίσταται υπεύθυνο για την ανάλυση των δεδομένων και την παραγωγή προτάσεων με βάση τα δεδομένα που ανακτούνται από το Hadoop. Επομένως, παίζει τον σημαντικότερο ρόλο στο σύστημα προτάσεων και για τον συγκεκριμένο λόγο θα πρέπει να προηγηθεί στοχευμένη μελέτη των προσφερόμενων τεχνολογιών που βοηθούν στη συγγραφή μιας διεργασίας του Spark που αφορά παραγωγή προτάσεων με γνώμονα πάντα τις απαιτήσεις της εφαρμογής.

Οι πιο δημοφιλείς γλώσσες που υποστηρίζουν το Apache Spark είναι οι Java, Python και Scala. Καθεμιά μπορεί εξίσου να υποστηρίξει το Spark μέσω βιβλιοθηκών ανοικτού κώδικα που έχουν αναπτυχθεί.

Γράφοντας μια διεργασία για το Spark που αφορά ανάλυση μεγάλου όγκου δεδομένων, θα πρέπει να λάβουμε υπ’ όψιν ότι ο προγραμματιστής θα πρέπει να είναι συνεχώς συγκεντρωμένος στον στόχο και όχι τόσο στον τρόπο με τον οποίο ο στόχος θα επιτευχθεί. Με απλά λόγια, τον προγραμματιστή θα πρέπει να τον ενδιαφέρει κυρίως να

πετύχει το επιθυμητό αποτέλεσμα της ανάλυσης του μεγάλου όγκου δεδομένων (παραγωγή προτάσεων) με τρόπο αποδοτικό και να μην επικεντρωθεί στον τρόπο με τον οποίο γίνεται η παραλληλοποίηση των δεδομένων και η ανάλυση τους σε χαμηλό επίπεδο.

Για τον άνωθεν λόγο, η γλώσσα Java θα πρέπει να αποκλειστεί. Αν και πολύ μεγάλων δυνατοτήτων, η Java είναι μια γλώσσα πολύλογη (verbose) και οι βιβλιοθήκες της δεν παρέχουν υψηλού επιπέδου υλοποιήσεις και κάτι τέτοιο θα ανάγκαζε τον προγραμματιστή να είναι περισσότερο συγκεντρωμένος στην υλοποίηση και όχι στο τελικό αποτέλεσμα. Η Java θα ήταν χρήσιμη αν η ανάλυση των δεδομένων γινόταν για άλλο σκοπό από την παραγωγή προτάσεων που είναι μια πιο πλαισιωμένη διαδικασία που βιβλιοθήκες υψηλότερου επιπέδου θα παρείχαν πιο απλές υλοποιήσεις και θα έκαναν πιο εύκολη και αποδοτική δουλειά για τον προγραμματιστή.

Επομένως, όπως είναι προφανές, οι επιλογές περιορίζονται στις γλώσσες Python και Scala. Είναι δύο γλώσσες που παρά τις διαφορές τους, παρέχουν ένα εξίσου υψηλού επιπέδου περιβάλλον για προγραμματισμό διεργασιών ανάλυσης μεγάλου όγκου δεδομένων με υλοποιήσεις του Spark στις προσφερόμενες βιβλιοθήκες τους. Οι απόψεις πάνω στο ποια από τις δύο αυτές γλώσσες είναι η επικρατέστερη διίστανται και ο διαχωρισμός των απόψεων βασίζεται κυρίως στο ποιο είναι το επιθυμητό αποτέλεσμα που θέλουμε να πετύχουμε με την ανάλυση των δεδομένων. Ας δούμε στην προκειμένη περίπτωση – σύστημα προτάσεων – ποια γλώσσα θα ήταν πιο ταιριαστή.

Πρώτα απ’ όλα, το πλαίσιο spark είναι γραμμένο σε γλώσσα προγραμματισμού Scala ως επί το πλείστον, οπότε η γλώσσα Scala αποκτά ένα προβάδισμα όσον αφορά την προτίμηση, καθώς θα υπάρχει όφελος σε ότι αφορά την επίδοση ενός προγράμματος γραμμένο σε Scala.

Η Python είναι μια γλώσσα λειτουργικού κώδικα (scripted language). Το συγκεκριμένο χαρακτηριστικό της την κάνει λιγότερο αποδοτική σε σύγκριση πάντα με τη γλώσσα Scala. Σε κάθε περίπτωση όμως, πρόκειται για μια γλώσσα εύκολη στη χρήση και πολύ υψηλού επιπέδου, επομένως η χρήση της θα έκανε πολύ πιο εύκολη τη δουλειά ενός προγραμματιστή [13].

Η γλώσσα Scala είναι μια νέα, σύγχρονη γλώσσα προγραμματισμού που συνδυάζει τα πλεονεκτήματα του αντικειμενοστραφούς προγραμματισμού με αυτά του συναρτησιακού. Η Scala προσφέρει μια ιεραρχία κλάσεων που βοηθά στην προτυποποίηση και στην επαναχρησιμοποίηση κώδικα, καθώς και συναρτήσεις υψηλού επιπέδου που κάνουν πολύ εύκολη την επεξεργασία περίπλοκων δομών χωρίς περιορισμό σε μέγεθος [14].

Το συμπέρασμα που βγαίνει από τα προαναφερόμενα είναι ότι η γλώσσα Python θα ήταν η ιδανική επιλογή για την εξοικείωση με το περιβάλλον του Spark και για την ανάπτυξη εφαρμογών σε δοκιμαστικό επίπεδο έως μέτριας πολυπλοκότητας. Η Scala από την άλλη, μπορεί να έχει αυξημένη πολυπλοκότητα σε σύγκριση με την Python, αλλά αποτελεί την ενδεδειγμένη επιλογή για εφαρμογές του Spark που η πολυπλοκότητά τους δεν μπορεί να πλαισιωθεί. Επομένως η γλώσσα που θα προτιμηθεί για το τμήμα της παραγωγής προτάσεων από το Spark είναι η Scala.

Αξίζει να τονισθεί ότι η άνωθεν συγκριτική μελέτη είναι έγκυρη μόνο για ότι αφορά τις απαιτήσεις της παρούσας ανάγκης για ανάλυση δεδομένων, της παραγωγής προτάσεων δηλαδή με τη χρήση βιβλιοθηκών του Spark. Διαφορετικού τύπου αναλύσεις δεδομένων θα μπορούσαν να επιδέχονται τη χρήση άλλων τεχνολογιών.

6.3 Κώδικας συστήματος προτάσεων

Όσον αφορά το σύστημα προτάσεων, αναπτύχθηκε κώδικας σε γλώσσα προγραμματισμού Java ώστε να γίνεται η επικοινωνία του συστήματος αρχείων του Hadoop με τη βάση δεδομένων μέσω διεργασίας map reduce και σε γλώσσα scala για την ανάλυση της αντλούμενης πληροφορίας και την παραγωγή προτάσεων μέσω του Spark.

6.3.1 Επικοινωνία MongoDB – HDFS μέσω διεργασίας map reduce

Για την επικοινωνία της βάσης με το σύστημα αρχείων του Hadoop συντάχθηκε σε γλώσσα προγραμματισμού Java μία διεργασία (process) map reduce που λαμβάνει την απαραίτητη πληροφορία από τη βάση δεδομένων. Παρακάτω ο κώδικας της διεργασίας η οποία εκτελείται από το εικονικό μηχάνημα.

```
1  import java.io.*;
2  import org.apache.hadoop.conf.*;
3  import org.apache.hadoop.fs.Path;
4  import org.apache.hadoop.io.*;
5  import org.apache.hadoop.mapreduce.lib.output.*;
6  import org.apache.hadoop.mapreduce.*;
7  import org.bson.*;
8  import com.mongodb.hadoop.*;
9  import com.mongodb.hadoop.util.*;
10
11 public class MongoHDFS {
12     public static class ReadWeblogsFromMongo extends Mapper<Object, BSONObject, Text, Text>{
13         // Create Map process to parse data from Mongo to HDFS
14         public void map(Object key, BSONObject value, Context context) throws IOException, InterruptedException{
15             String AuthorNum =  String.valueOf(value.get("AuthorNum"));
16             String Hotel = (String) value.get("Hotel");
17             Object Ratings = value.get("Ratings");
18             String OverallRating = (String) ((BSONObject) Ratings).get("Overall");
19             String Data = Hotel + "\t" + OverallRating;
20
21             context.write( new Text(AuthorNum), new Text(Data));
22         }
23     }
24
25     public static void main(String[] args) throws Exception{
26         final Configuration conf = new Configuration();
27
28         // Connection to mongodb from vm to host machine
29         MongoConfigUtil.setInputURI(conf,"mongodb://192.168.1.80:27017/tripadvisorDB.Review");
30         MongoConfigUtil.setCreateInputSplits(conf, false);
31
32         // Run the mapping process
33         final Job job = new Job(conf, "Mongo Import");
34
35         // Path to HDFS
36         Path out = new Path("/tripadvisor/reviews/mongo_import");
37         FileOutputFormat.setOutputPath(job, out);
38
39         job.setJarByClass(MongoHDFS.class);
40         job.setMapperClass(ReadWeblogsFromMongo.class);
41         job.setOutputKeyClass(Text.class);
42         job.setOutputValueClass(Text.class);
43         job.setInputFormatClass(MongoInputFormat.class);
44         job.setOutputFormatClass(TextOutputFormat.class);
45         job.setNumReduceTasks(0);
46
47         System.exit(job.waitForCompletion(true) ? 0 : 1 );
48     }
49 }
```

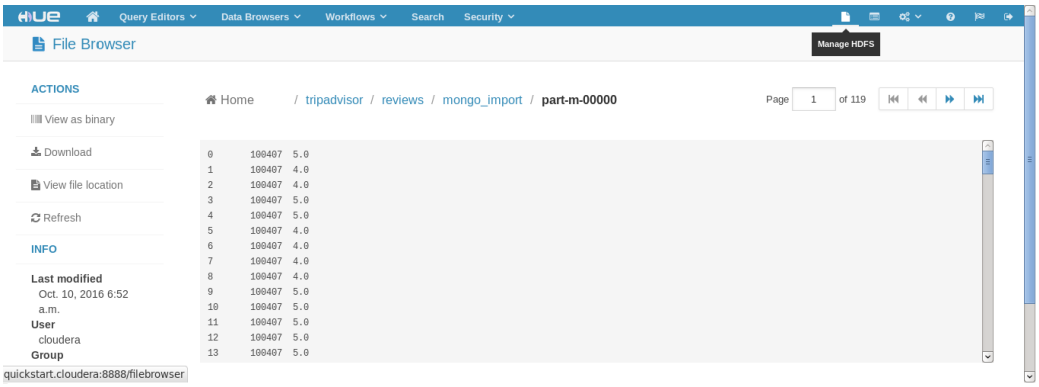
Εικόνα 17: Κώδικας διεργασίας map reduce

Πρακτικά, η δουλειά της διεργασίας είναι να συνδέεται στη βάση δεδομένων που βρίσκεται στο μηχάνημα host (το μηχάνημα που φιλοξενεί το εικονικό μηχάνημα δηλαδή) (γραμμές 29-30) και να συλλέγει τα απαραίτητα δεδομένα, δημιουργώντας διεργασία τύπου map reduce (γραμμή 33) για να τα αποθηκεύσει στο σύστημα αρχείων του Hadoop. Ουσιαστικά εδώ χρησιμοποιείται κυρίως η δυνατότητα mapping, αφού δεν χρειάζεται να γίνει κάποια ανάλυση των δεδομένων, απλά αποθηκεύονται στο σύστημα αρχείων του Hadoop (γραμμή 36).

Στην περίπτωσή μας, από τη συλλογή των κριτικών που είναι αποθηκευμένες στη βάση δεδομένων, κρατά το αναγνωριστικό κριτικού, το ξενοδοχείο πάνω στο οποίο έγινε η

κριτική και τη βαθμολογία που δόθηκε. Τα 3 αυτά στοιχεία είναι αρκετά ώστε να δουλέψει στην πορεία η διαδικασία του αλγορίθμου Collaborative Filtering όπως αυτή προσφέρεται σαν API μέσα από τις βιβλιοθήκες του Spark σε γλώσσα προγραμματισμού Scala.

Το αποτέλεσμα του παραπάνω κώδικα φαίνεται στην παρακάτω εικόνα, που μέσω του εργαλείου Hue της cloudera, δίνει μια εικόνα του συστήματος αρχείων του Hadoop.



Εικόνα 18: Αποτελέσματα εκτέλεσης της διεργασίας map reduce

Όπως προαναφέρθηκε, η πρώτη στήλη αφορά το αναγνωριστικό κριτικού, η δεύτερη στήλη τον κωδικό του ξενοδοχείου και η τελευταία στήλη τη βαθμολογία που δόθηκε από τον κριτικό στο ξενοδοχείο.

6.3.2 Ανάλυση δεδομένων με τη χρήση του Apache Spark

Έπειτα από την άτληση της πληροφορίας και την αποθήκευσή της στο σύστημα αρχείων του Hadoop μέσω διεργασίας map reduce, θα αναλάβει το λογισμικό Spark με χρήση του ALS αλγορίθμου να εξάγει προβλέψεις και προτάσεις του τύπου:

(αναγνωριστικό κριτικού, κωδικός ξενοδοχείου , βαθμολογία)

και μέσω της βιβλιοθήκης org.mongodb.scala, θα γίνει και η αποθήκευση των προτάσεων στη βάση δεδομένων.

Για τις ανάγκες της βιβλιοθήκης mlib του Spark, απαιτείται να παρέχουμε στο σύστημα 3 βασικές παραμέτρους:

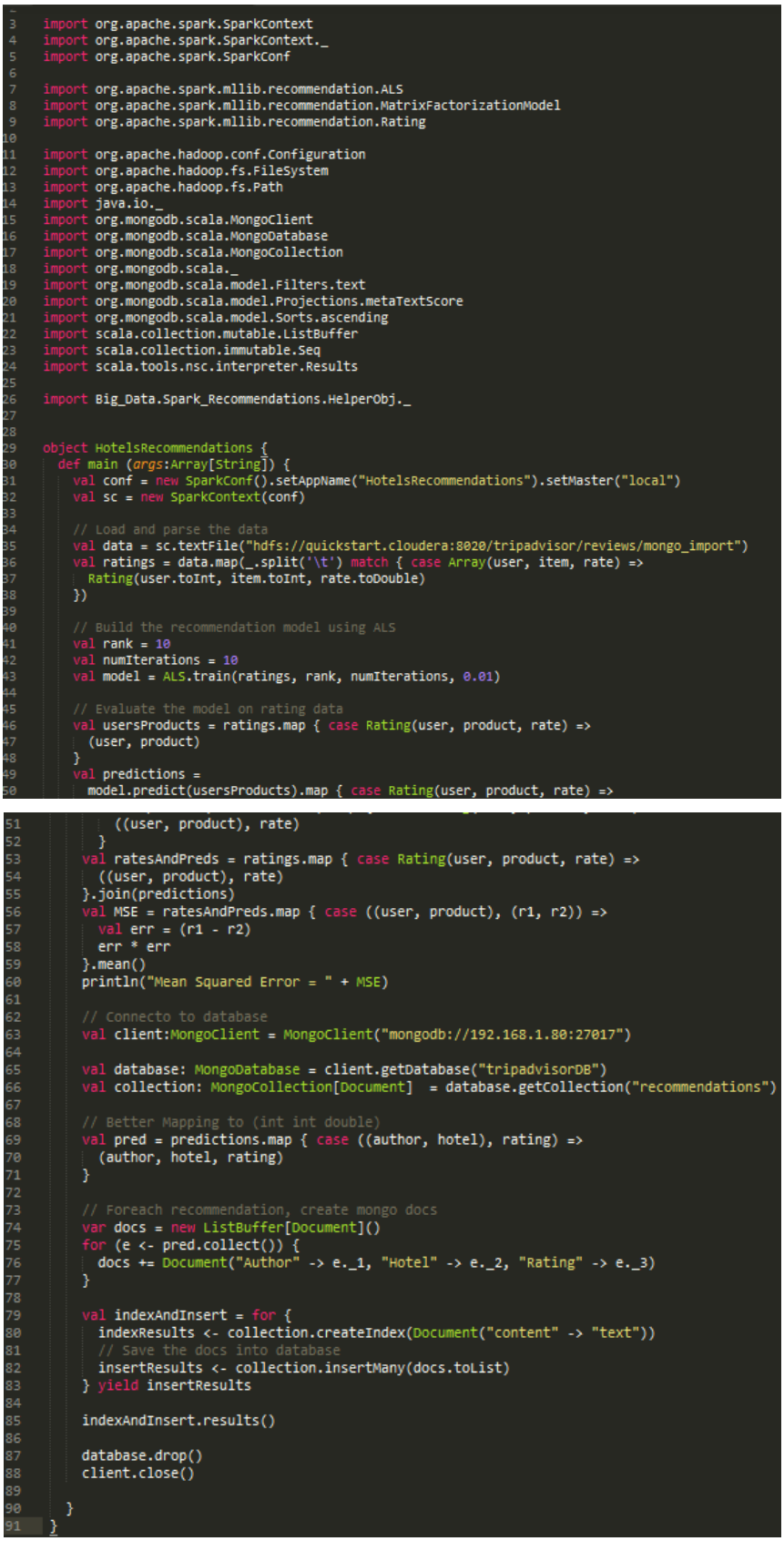
- Αναγνωριστικό Χρήστη (user)
- Αναγνωριστικό Αντικειμένου (item)
- Βαθμολογία Χρήστη στο Αντικείμενο

Όπως είναι προφανές, στο συγκεκριμένο παράδειγμα, οι χρήστες δεν είναι άλλοι από τους κριτικούς ξενοδοχείων, τα αντικείμενα είναι τα ξενοδοχεία και η βαθμολογία είναι αυτή που έδωσαν οι κριτικοί στα ξενοδοχεία.

Αφότου δοθούν οι κατάλληλες παράμετροι στη βιβλιοθήκη mlib, μέσω της μεθόδου ALS.predict της ίδιας βιβλιοθήκης, παράγονται προτάσεις ξενοδοχείων υπό τη μορφή tuples του τύπου:

((αναγνωριστικό κριτικού, κωδικός ξενοδοχείου) , βαθμολογία)

Στη συνέχεια, η παραπάνω μορφή tuple απλοποιείται στη μορφή (αναγνωριστικό κριτικού, κωδικός ξενοδοχείου , βαθμολογία) ώστε να γίνει η σύνδεση με τη βάση δεδομένων και να αποθηκευτούν οι προτάσεις με τη χρήση της βιβλιοθήκης org.mongodb.scala της γλώσσας scala,όπως φαίνεται και στην ακόλουθη εικόνα.



```scala
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._
import org.apache.spark.SparkConf

import org.apache.spark.mllib.recommendation.ALS
import org.apache.spark.mllib.recommendation.MatrixFactorizationModel
import org.apache.spark.mllib.recommendation.Rating

import org.apache.hadoop.conf.Configuration
import org.apache.hadoop.fs.FileSystem
import org.apache.hadoop.fs.Path
import java.io._
import org.mongodb.scala.MongoClient
import org.mongodb.scala.MongoDatabase
import org.mongodb.scala.MongoCollection
import org.mongodb.scala._
import org.mongodb.scala.model.Filters.text
import org.mongodb.scala.model.Projections.metaTextScore
import org.mongodb.scala.model.Sorts.ascending
import scala.collection.mutable.ListBuffer
import scala.collection.immutable.Seq
import scala.tools.nsc.interpreter.Results

import Big_Data.Spark_Recommendations.HelperObj._


object HotelsRecommendations {
  def main (args:Array[String]) {
    val conf = new SparkConf().setAppName("HotelsRecommendations").setMaster("local")
    val sc = new SparkContext(conf)

    // Load and parse the data
    val data = sc.textFile("hdfs://quickstart.cloudera:8020/tripadvisor/reviews/mongo_import")
    val ratings = data.map(_.split('\t') match { case Array(user, item, rate) =>
      Rating(user.toInt, item.toInt, rate.toDouble)
    })

    // Build the recommendation model using ALS
    val rank = 10
    val numIterations = 10
    val model = ALS.train(ratings, rank, numIterations, 0.01)

    // Evaluate the model on rating data
    val usersProducts = ratings.map { case Rating(user, product, rate) =>
      (user, product)
    }
    val predictions =
      model.predict(usersProducts).map { case Rating(user, product, rate) =>
        ((user, product), rate)
      }
    val ratesAndPreds = ratings.map { case Rating(user, product, rate) =>
      ((user, product), rate)
    }.join(predictions)
    val MSE = ratesAndPreds.map { case ((user, product), (r1, r2)) =>
      val err = (r1 - r2)
      err * err
    }.mean()
    println("Mean Squared Error = " + MSE)

    // Connecto to database
    val client:MongoClient = MongoClient("mongodb://192.168.1.80:27017")

    val database: MongoDatabase = client.getDatabase("tripadvisorDB")
    val collection: MongoCollection[Document]  = database.getCollection("recommendations")

    // Better Mapping to (int int double)
    val pred = predictions.map { case ((author, hotel), rating) =>
      (author, hotel, rating)
    }

    // Foreach recommendation, create mongo docs
    var docs = new ListBuffer[Document]()
    for (e <- pred.collect()) {
      docs += Document("Author" -> e._1, "Hotel" -> e._2, "Rating" -> e._3)
    }

    val indexAndInsert = for {
      indexResults <- collection.createIndex(Document("content" -> "text"))
      // Save the docs into database
      insertResults <- collection.insertMany(docs.toList)
    } yield insertResults

    indexAndInsert.results()

    database.drop()
    client.close()

  }
}
```

Εικόνα 19: Κώδικας σε γλώσσα scala για το Apache Spark

Όπως φαίνεται και μέσα από τον κώδικα, το Spark θα χρειαστεί την πληροφορία που προηγουμένως αποθηκεύτηκε στο σύστημα αρχείων του Hadoop (γραμμές 35 -37) ώστε να δημιουργήσει τα κατάλληλα αντικείμενα (objects) που θα χρησιμοποιηθούν για την εκτέλεση των παρακάτω μεθόδων και την εφαρμογή του collaborative filtering της γλώσσα scala (γραμμές 41 - 60).

Στη συνέχεια, γίνεται σύνδεση με τη βάση δεδομένων και αποθήκευση των αποτελεσμάτων σε αυτή μέσω της βιβλιοθήκης org.mongodb.scala. Αφότου πετύχει η σύνδεση με τη βάση δεδομένων και ληφθεί η συλλογή που μας ενδιαφέρει (γραμμές 63 - 66), απλοποιούμε τη μορφή της λίστας από tuples (γραμμές 69 - 71) που προαναφέρθηκε και αντλούμε την πληροφορία η οποία θα αποθηκευτεί στη βάση δεδομένων υπό τη μορφή αρχείων της mongodb (γραμμές 74 - 77). Τέλος απομένει η αποθήκευση των προτάσεων στη βάση (γραμμές 79 - 85) και ο τερματισμός της σύνδεσης με τη βάση (γραμμές 87 - 88).

Για την εκτέλεση του παραπάνω τμήματος κώδικα, δημιουργήθηκε ένα Maven Project με εξαρτήσεις (dependencies) της γλώσσας Scala.

Η παραπάνω εφαρμογή αποθηκεύει τα αποτελέσματα των προτάσεων στη βάση δεδομένων ως εξής:

```
> db.recommendations.find().sort({Author:1}).pretty()
{
        "_id" : ObjectId("586bf838bb58ca25c1e334a0"),
        "Author" : 0,
        "Hotel" : 100407,
        "Rating" : 4.99453295715225
}
{
        "_id" : ObjectId("586bf838bb58ca25c1e331d1"),
        "Author" : 1,
        "Hotel" : 100504,
        "Rating" : 4.990806932192288
}
{
        "_id" : ObjectId("586bf838bb58ca25c1e3348b"),
        "Author" : 1,
        "Hotel" : 100407,
        "Rating" : 3.997118733446351
}
```

Εικόνα 20: Αποτελέσματα προτάσεων στη βάση μετά την εκτέλεση της διεργασίας του Spark

Παρατηρούμε λοιπόν, ότι με τη χρήση των δυνατοτήτων που προσφέρει το Spark, μπορεί ο προγραμματιστής έχοντας σαν κύριο στόχο το αποτέλεσμα (παραγωγή προτάσεων), μέσω μικρής έκτασης κώδικα, να φτάσει στον στόχο αυτό χωρίς να πρέπει να μεριμνήσει για τη διαδικασία μέσω της οποίας θα επιτύχει το σκοπό του. Κάτι τέτοιο γίνεται εφικτό από τη στιγμή που η διαδικασία εφαρμογής του αλγορίθμου Collaborative Filtering έχει πλαισιωθεί υπό πολλές παραλλαγές του μέσω της βιβλιοθήκης mlib του Spark.

7. ΣΥΜΠΕΡΑΣΜΑΤΑ

Είναι σαφές ότι με την πάροδο των χρόνων και την ραγδαία ανάπτυξη που γνωρίζει ο χώρος των διαδικτυακών εφαρμογών, ολοένα και νεότερες – πιο σύγχρονες τεχνολογίες αναπτύσσονται και αντικαθιστούν τις ήδη υπάρχουσες. Ταυτόχρονα με την πολυπλοκότητα και τις απαιτήσεις των χρηστών, αυξάνει και ο όγκος των δεδομένων που μια διαδικτυακή εφαρμογή καλείται να αντιμετωπίσει. Προκειμένου μια εφαρμογή να λειτουργεί αποδοτικά ενώ ο όγκος των δεδομένων που διαχειρίζεται αυξάνει συνεχώς με ραγδαίους ρυθμούς, θα πρέπει να υπάρξει μέριμνα από την ομάδα προγραμματιστών ώστε ο όγκος δεδομένων να γίνει ευκολότερα διαχειρίσιμος. Για τον συγκεκριμένο σκοπό, έχουν αναπτυχθεί νέα πλαίσια λογισμικού και νέες αρχιτεκτονικές εφαρμογών διαδικτύου που μπορούν να χρησιμοποιηθούν προκειμένου να επιτυγχάνεται το επιθυμητό αποτέλεσμα.

Η εφαρμογή που αναπτύχθηκε είχε ως σκοπό την ενασχόληση με διαδικτυακές εφαρμογές που αντιμετωπίζουν δεδομένα μεγάλου όγκου και τις δυσκολίες που παρουσιάζονται στη διαχείρισή τους από αυτές.

Γενικώς, για τα πλαίσια λογισμικού που προτιμήθηκαν τόσο σε επίπεδο ανάπτυξης της διαδικτυακής πλατφόρμας, όσο και σε επίπεδο ανάλυσης δεδομένων υπήρξε μέριμνα ώστε να είναι ειδικά προσανατολισμένα στις απαιτήσεις για τον προγραμματισμό της εφαρμογής. Χαρακτηριστικά, χρησιμοποιήθηκε το Strongloop για την ανάπτυξη του REST Api αφού είναι ένα πλαίσιο λογισμικού ειδικά προσανατολισμένο για αυτόν τον σκοπό. Το ίδιο συνέβη και με το τμήμα της ανάλυσης δεδομένων, αφού επιλέχθηκαν πλαίσια λογισμικού που έχουν αναπτυχθεί με πλαισιωμένες λειτουργίες που παρέχουν έτοιμες λειτουργίες για βασικές τεχνικές ανάλυσης δεδομένων. Γίνεται προφανές λοιπόν ότι η χρήση των κατάλληλων πλαισίων λογισμικού διευκόλυνε τη διαδικασία της ανάπτυξης του κώδικα, χωρίς να χρειάζεται να «ανακαλύπτουμε τον τροχό» για κάθε λειτουργία που αναπτυσσόταν.

Λόγω των υψηλών υπολογιστικών απαιτήσεων που έχουν δεδομένα μεγάλου όγκου, το τμήμα της ανάλυσής τους θα πρέπει να γίνεται με σύγχρονα μέσα, όπως και πραγματοποιήθηκε χρησιμοποιώντας μια μη σχεσιακή βάση, τη MongoDB, σε συνδυασμό με το οικοσύστημα του Hadoop και του Spark, δύο πλαισίων λογισμικού προσανατολισμένα ακριβώς γι’ αυτό τον σκοπό. Αν και όπως προαναφέρθηκε τα δύο αυτά πλαίσια είναι προσανατολισμένα για τη συγκεκριμένη δουλειά, το γεγονός ότι βασικές τους λειτουργίες είναι ειδικά προσαρμοσμένες, απαιτεί μια καλή εξοικείωση με αυτά, καθώς νέες έννοιες και τεχνικές πρέπει να χρησιμοποιούνται (πχ η έννοια του map reduce για το Hadoop). Η ενασχόληση με τις νέες αυτές τεχνολογίες σε συνδυασμό με την υποστήριξη από τις ομάδες προγραμματιστών που τις αναπτύσσουν, καθιστά ευκολότερη την εξοικείωση με αυτές και κατ’ επέκταση γρηγορότερη την εκμάθηση των νέων τεχνικών που πρέπει να ακολουθούνται.

Ακόμη, με τη χρήση μιας μη σχεσιακής βάσης δεδομένων, έχοντας εμπειρία μόνο πάνω σε σχεσιακές, προκύπτουν προβλήματα ως προς τις βασικές αρχές που πρέπει να ακολουθούνται. Με τη μελέτη της φύσης των μη σχεσιακών βάσεων δεδομένων, και ιδιαίτερα της MongoDB που παρέχει μια αρκετά εύχρηστη διεπαφή χρήστη, τα προβλήματα που προκύπτουν είναι δυνατόν να αντιμετωπιστούν ευκολότερα.

Τέλος, η ανάπτυξη ενός REST Api με τη χρήση του Strongloop και η σύνδεσή του με ένα εμπρόσθιο μέρος ανεπτυγμένο σε Angular.js πραγματοποιήθηκε με επιτυχία, αξιοποιώντας τις μεγάλες δυνατότητες που παρέχουν τα δύο αυτά πλαίσια.

Η ανάπτυξη μιας διαδικτυακής εφαρμογής με τη χρήση των τεχνολογιών που παρουσιάστηκαν και ακολουθώντας σύγχρονα μοντέλα αρχιτεκτονικών ανάπτυξης

διαδικτυακών εφαρμογών είναι μια αρκετά επωφελής διαδικασία. Πέραν της εμπειρίας που αποκτά με τη μελέτη και χρήση των συγκεκριμένων τεχνολογιών, ανοίγονται στον προγραμματιστή νέοι ορίζοντες που τον εισάγουν στη νέα γενιά της ανάπτυξης διαδικτυακών εφαρμογών.

Πιο συγκεκριμένα, στο τμήμα της ανάπτυξης μιας διαδικτυακής εφαρμογής όπως της παρούσας, η γλώσσα Javascript που αποτελεί τον ακρογωνιαίο λίθο του προγραμματισμού διαδικτυακών εφαρμογών, έχει γνωρίσει μεγάλη άνθηση τα τελευταία χρόνια. Για την ακρίβεια, σχεδόν για κάθε ανάγκη που γεννιέται κατά την ανάπτυξη μιας εφαρμογής διαδικτύου, έχει αναπτυχθεί και το κατάλληλο πλαίσιο λογισμικού. Ταυτόχρονα, σημαντική λεπτομέρεια είναι και το γεγονός ότι έχουμε φτάσει στο σημείο να μπορούμε να χρησιμοποιήσουμε την γλώσσα Javascript ακόμη και για τον προγραμματισμό του συστήματος υποστήριξης μιας εφαρμογής, όπως παρουσιάστηκε στην παρούσα εργασία με τη χρήση της node.js και των πλαισίων λογισμικού της (Loopback - Strongloop).

Όσον αφορά το εμπρόσθιο τμήμα της εφαρμογής, η Angular.js, ένα σύγχρονο πλαίσιο λογισμικού της Javascript, παρέχει μεγάλες δυνατότητες για ανάπτυξη εφαρμογών και η χρήση της, πέραν της ευκολίας που παρέχει στην ανάπτυξη σύγχρονων διεπαφών χρήστη, βοηθά τον προγραμματιστή να αποκτά εμπειρία ανάπτυξης μοντέρνων πλατφόρμων με αποδοτικό και εύκολο τρόπο. Ακόμη, διδάσκει νέες τεχνικές προγραμματισμού του εμπρόσθιου τμήματος της εφαρμογής (πχ αμφίδρομη αντιστοίχιση και routing) που είναι πλέον επιτακτικές για την ανάπτυξη σύγχρονων διαδικτυακών εφαρμογών.

Επίσης, με τη χρήση της υπηρεσιοστραφούς αρχιτεκτονικής και των RESTful διαδικτυακών υπηρεσιών, εδραιώνεται η επικοινωνία του εμπρόσθιου τμήματος με το σύστημα υποστήριξης και με τη χρήση των πλαισίων λογισμικού της Javascript που παρουσιάστηκαν, γίνεται αρκετά απλή η ανάπτυξη ενός REST API που παρέχεται για την επικοινωνία των προγραμμάτων πελάτη και εξυπηρετητή.

Η ανάπτυξη ενός REST API με τη χρήση του Strongloop πραγματοποιείται εύκολα, γρήγορα και αποδοτικά. Επομένως παρατηρούμε ότι η εισαγωγή της Javascript στο σύστημα υποστήριξης, μέσω της node.js, με τη χρήση του πλαισίου Strongloop, κάνει την προσπάθεια για την ανάπτυξη ενός API πολύ μικρότερη απ’ ότι θα ήταν με τη χρήση άλλων τεχνολογιών. Επομένως ο προγραμματιστής που θα αναπτύξει ένα REST API με τη χρήση του Strongloop για την εφαρμογή του, θα εξοικειωθεί με τη νέα αρχιτεκτονική ανάπτυξης εφαρμογών εύκολα .

Χρησιμοποιώντας το πλαίσιο Strongloop σε συνδυασμό με Angular.js, επιτυγχάνεται ομαλή ροή των δεδομένων. Ένα πρόγραμμα πελάτη γραμμένο σε Angular.js κάνοντας τα κατάλληλα αιτήματα στο REST Api που αναπτύχθηκε, εξασφαλίζει άμεσα και επιτυχώς την επικοινωνία με τη βάση δεδομένων. Το ίδιο το REST Api είναι ανεπτυγμένο ώστε να μπορεί να επικοινωνεί με οποιοδήποτε πρόγραμμα πελάτη, γραμμένο σε οποιαδήποτε γλώσσα προγραμματισμού αρκεί να γίνονται τα σωστά αιτήματα.

Όσον αφορά τα δεδομένα της εφαρμογής – τα Μεγάλα Δεδομένα, εισάγοντας την τεχνολογία μιας μη σχεσιακής αρχειοστραφούς βάσης δεδομένων (MongoDB), είμαστε σε θέση να πετύχουμε την καλύτερη δυνατή ομαδοποίηση των αδόμητων δεδομένων με πιο αποδοτικό τρόπο. Χρησιμοποιώντας μια μη σχεσιακή βάση δεδομένων, όπως τη MongoDB, ο προγραμματιστής έρχεται σε επαφή με μια νέα μορφή βάσεων δεδομένων και με νέες έννοιες που εισάγονται όλο και περισσότερο στον τομέα της σχεδίασης βάσεων δεδομένων (πχ αρχειοστραφές μοντέλο).

Τέλος, σχετικά με την ανάλυση των μεγάλων δεδομένων της βάσης, χρησιμοποιώντας τα πλαίσια λογισμικού της Apache, το Hadoop και το Spark, που τρέχουν ανεξάρτητα από την εφαρμογή σε εικονικό μηχάνημα, αναπτύχθηκε ένα παράδειγμα συστήματος προτάσεων που κρατά τη βάση δεδομένων ενήμερη με προτάσεις αντικειμένων (ξενοδοχείων στην προκειμένη). Τα συγκεκριμένα πλαίσια λογισμικού δίνουν πολύ μεγάλη υπολογιστική ισχύ και ασφάλεια για τα δεδομένα που επεξεργάζονται.

Τα πλαίσια λογισμικού Hadoop και Spark για την ανάλυση δεδομένων μεγάλου όγκου με τη χρήση εικονικού μηχανήματος, προσφέρουν μια νέα διαφορετική εμπειρία στον τομέα της ανάλυσης δεδομένων. Ο προγραμματιστής επικεντρωμένος πλέον κυρίως στον στόχο του (παραγωγή προτάσεων στην προκειμένη) και όχι στη διαδικασία της ανάπτυξης κώδικα ώστε να τον πετύχει, μπορεί με τη χρήση των βιβλιοθηκών του Spark (πχ βιβλιοθήκη mlib) να καταναλώνεται λιγότερο στο «πως» και περισσότερο στο «τι» θέλει να δημιουργήσει. Η γλώσσα Scala που χρησιμοποιήθηκε για τον συγκεκριμένο σκοπό είναι μια αρκετά σύγχρονη γλώσσα αρκετά υψηλού επιπέδου που παρέχει ευκολία στον προγραμματιστή να αναπτύσσει περίπλοκες εφαρμογές χωρίς να είναι αναγκασμένος να γράφει μεγάλα κομμάτια κώδικα. Πρόκειται για μια γλώσσα που συνδυάζει τα πλεονεκτήματα που παρέχει ο συναρτησιακός προγραμματισμός με αυτά του αντικειμενοστραφούς προγραμματισμού και επομένως, η χρήση της μπορεί να διδάξει νέες τεχνικές προγραμματισμού αρκετά χρήσιμες για την ανάπτυξη ισχυρών εφαρμογών.

Το σκεπτικό με βάση το οποίο πρόκειται να λειτουργήσει το σύστημα προτάσεων έχει ως εξής:

• Ανεξάρτητα από το αν η εφαρμογή βρίσκεται σε μορφή online ή offline, ανά τακτά χρονικά διαστήματα, θα τρέχουν οι διεργασίες που θα αντλούν την απαραίτητη πληροφορία από τη βάση δεδομένων και θα την τοποθετούν στο σύστημα αρχείων του Hadoop προς ανάλυση.

• Αμέσως μετά την άντληση της πληροφορίας, θα αναλαμβάνει η διεργασία που αναλύοντας την πληροφορία αυτή θα παράγει αποτελέσματα προτάσεων ξενοδοχείων, όπως περιγράφονται παραπάνω.

• Αφότου παραχθούν οι προτάσεις ξενοδοχείων, θα πρέπει αυτές να αποθηκευτούν πίσω στη βάση δεδομένων

Με την παραπάνω διαδικασία να επαναλαμβάνεται ανά τακτά χρονικά διαστήματα (πχ 1 με 2 φορές ανά 24ωρο), θα μπορούσε να υπάρχει συνεχής ενημέρωση της βάσης με έγκυρες προτάσεις ξενοδοχείων για κάθε χρήστη, χωρίς να υπάρχει καμία μέριμνα από πλευράς ομάδας προγραμματιστών της εφαρμογής. Δηλαδή, ανεξάρτητα από το αν η εφαρμογή βρίσκεται online ή offline ή από το αν βρίσκεται σε στάδιο ανάπτυξης, η ομάδα προγραμματιστών του συστήματος υποστήριξης μπορεί να εργάζεται ανεξάρτητα χωρίς να υπάρχει η οποιαδήποτε ανησυχία για απώλεια δεδομένων ή μείωση της απόδοσης της εφαρμογής.

Επιλογικά, όσο περνούν τα χρόνια, νέες τεχνολογίες εφαρμογών διαδικτύου θα συνεχίζουν να αναπτύσσονται με σκοπό την κάλυψη των αναγκών που προκύπτουν από τις απαιτήσεις των χρηστών και την αυξανόμενη πολυπλοκότητα τόσο των ίδιων των εφαρμογών, όσο και των δεδομένων που θα πρέπει να επεξεργάζονται. Επομένως, είναι επιτακτική η ανάγκη της συστηματικής μελέτης των νέων τεχνολογιών που εμφανίζονται, καθώς και της σωστής κρίσης στην επιλογή των καταλληλότερων βάσει πάντα των αναγκών της εφαρμογής που αναπτύσσεται. Η ανάπτυξη της παρούσας

εφαρμογής παρουσιάζει τις βασικές αρχές των τεχνολογιών που αναλύθηκαν, προσφέροντας ένα απλό παράδειγμα της ορθής χρήσης τους. Κατά τη διαδικασία της ανάπτυξης, αποκτήθηκε σημαντική εμπειρία πάνω στο αντικείμενο της ανάπτυξης Εφαρμογών Διαδικτύου και στη συνύπαρξή τους με μεγάλο όγκο δεδομένων. Εμπειρία που είναι απαραίτητη για την εισαγωγή στη νέα γενιά ανάπτυξης εφαρμογών και ανάλυσης δεδομένων με σύγχρονους τρόπους και σύγχρονες τεχνικές προγραμματισμού.

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος όρος	**Ελληνικός Όρος**
Big Data	Μεγάλα Δεδομένα
unstructured data	Αδόμητα Δεδομένα
Relational Databases	Σχεσιακές Βάσεις Δεδομένων
Non Relational Databases	Μη Σχεσιακές Βάσεις Δεδομένων
frameworks	Πλαίσια λογισμικού
scalable	επεκτάσιμος
document oriented	Αρχειοστραφής
Document	Αρχείο
Row	Πλειάδα
Object	Αντικείμενο
nested object	Εμφωλευμένο αντικείμενο
collections	Συλλογές Δεδομένων
Web Services	Διαδικτυακές Υπηρεσίες
Server	Εξυπηρετητής
Client	Πελάτης
backend	Σύστημα Υποστήριξης εφαρμογής
Front end	Εμπρόσθιο Τμήμα εφαρμογής
Create	Δημιουργία
Read	Προσπέλαση
Update	Ενημέρωση
Delete	Διαγραφή
Trends	Τάσεις
templating	προτυποποίηση
dependences	εξαρτήσεις
Component	στοιχείο
User interface	Διεπαφή χρήστη
Binding	Αντιστοίχιση
Element	Στοιχείο
View	Όψη
Model	Μοντέλο
two – way data binding	Αμφίδρομη αντιστοίχιση
Services	Υπηρεσίες
Instance	Στιγμιότυπο
Debugging	Αποσφαλμάτωση
Business Logic	Λογική της Εφαρμογής
Modules	Δομές
Queries	Ερωτήματα
Horizontal scale	Οριζόντια διαβάθμιση
Real time processing	Επεξεργασία σε πραγματικό χρόνο
Clusters	Υποσύνολο δεδομένων
Processing time	Μέσος χρόνος επεξεργασίας
Recommendation System	Σύστημα Προτάσεων
Namenode	Κεντρικός Κόμβος
Datanode	Κόμβος Επεξεργασίας Δεδομένων
Hardware	Υλικό
Network failures	Αποτυχίες δικτύου
Dataset	Σύνολο δεδομένων
Virtual Machine	Εικονικό μηχάνημα

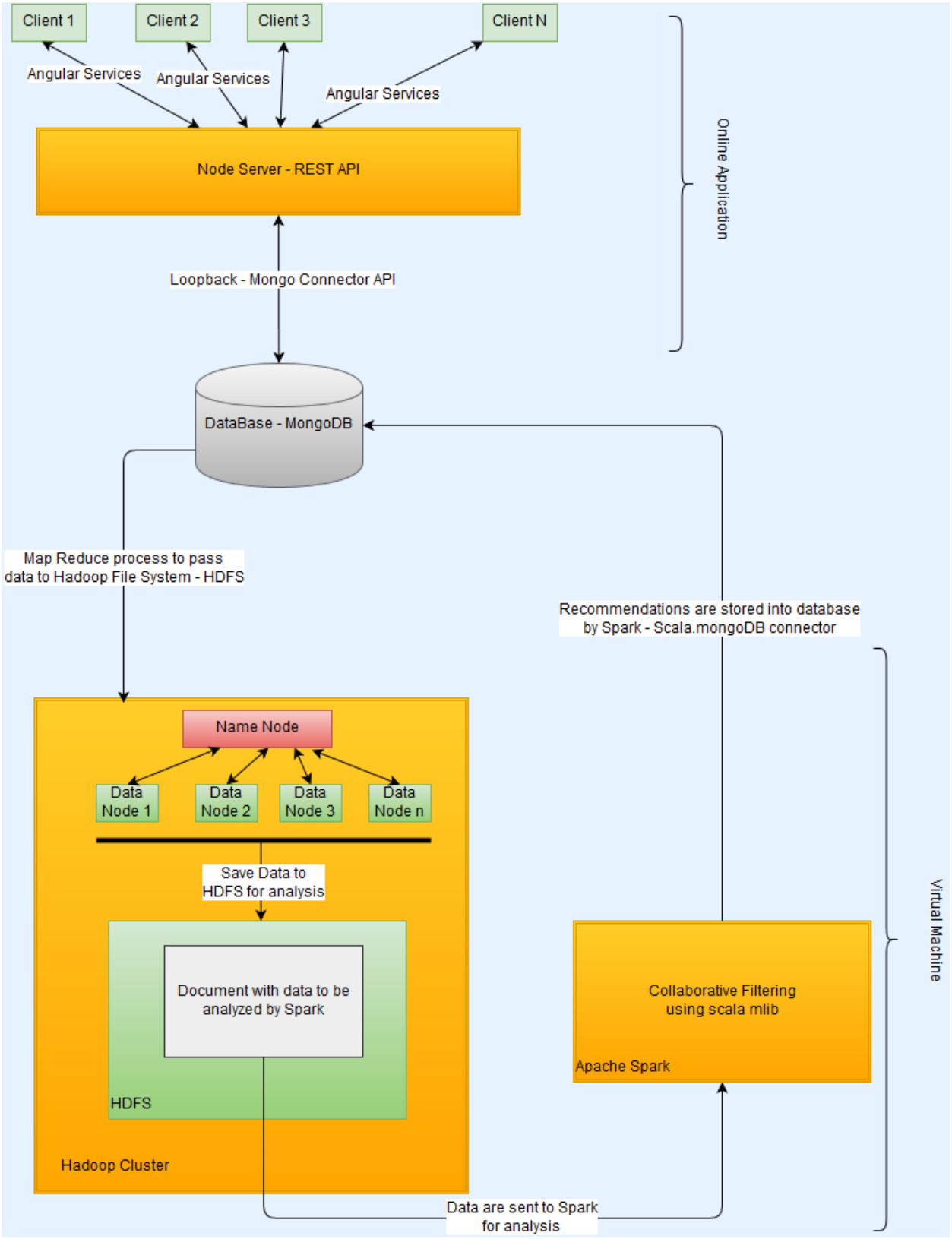
Image	Εικόνα
Master Namenode	Κύριος Κόμβος
Worker Node	Κόμβος Εργάτης
Verbose	Πολύλογη γλώσσα προγραμματισμού
Scripted Language	Γλώσσα Λειτουργικού Κώδικα
Process	Διεργασία
Rating	Βαθμολογία

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

JSON	Javascript Object Notation
BSON	Binary Structured Object Notation
REST	Representational State Transfer
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
WSDL	Web *Services* Description Language
HTTP	Hyper Text Transfer Protocol
SPAs	Single Page Apps
MVC	Model View Controller
HTML	HyperText Markup Language
HDFS	HaDoop File System

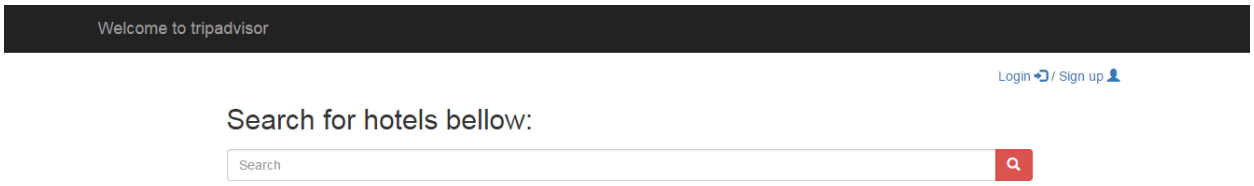
ΠΑΡΑΡΤΗΜΑ Ι

Στο ακόλουθο σχήμα διακρίνεται η αρχιτεκτονική της εφαρμογής

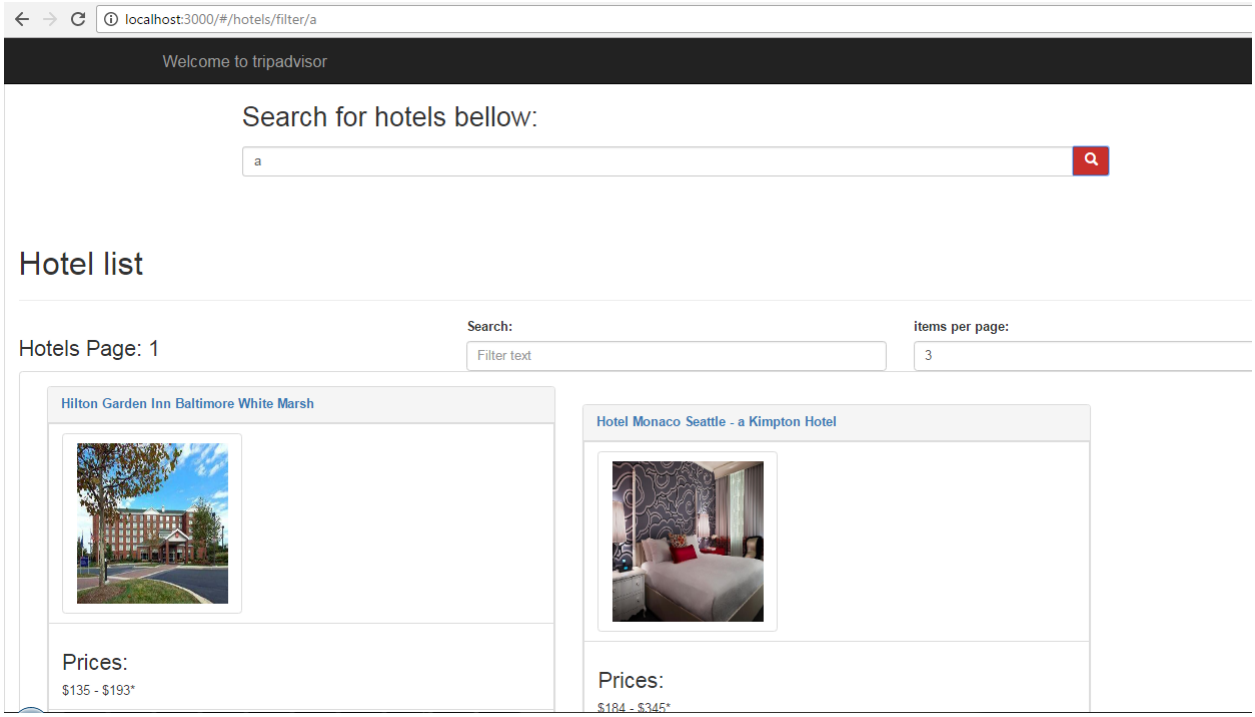


Σχήμα 4: Ανατομία της εφαρμογής

ΠΑΡΑΡΤΗΜΑ ΙΙ



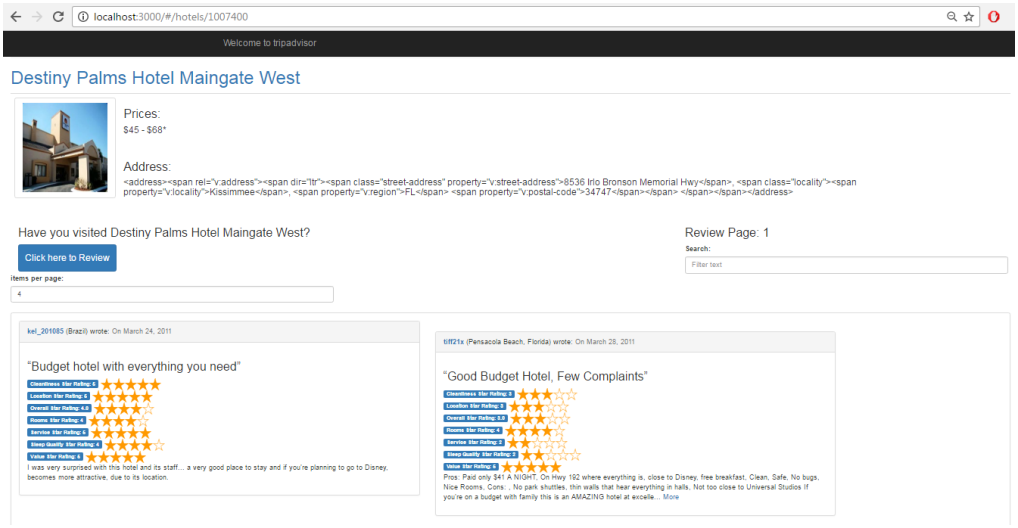
Εικόνα 21: Αρχική σελίδα εφαρμογής



Εικόνα 22: Αποτελέσματα αναζήτησης

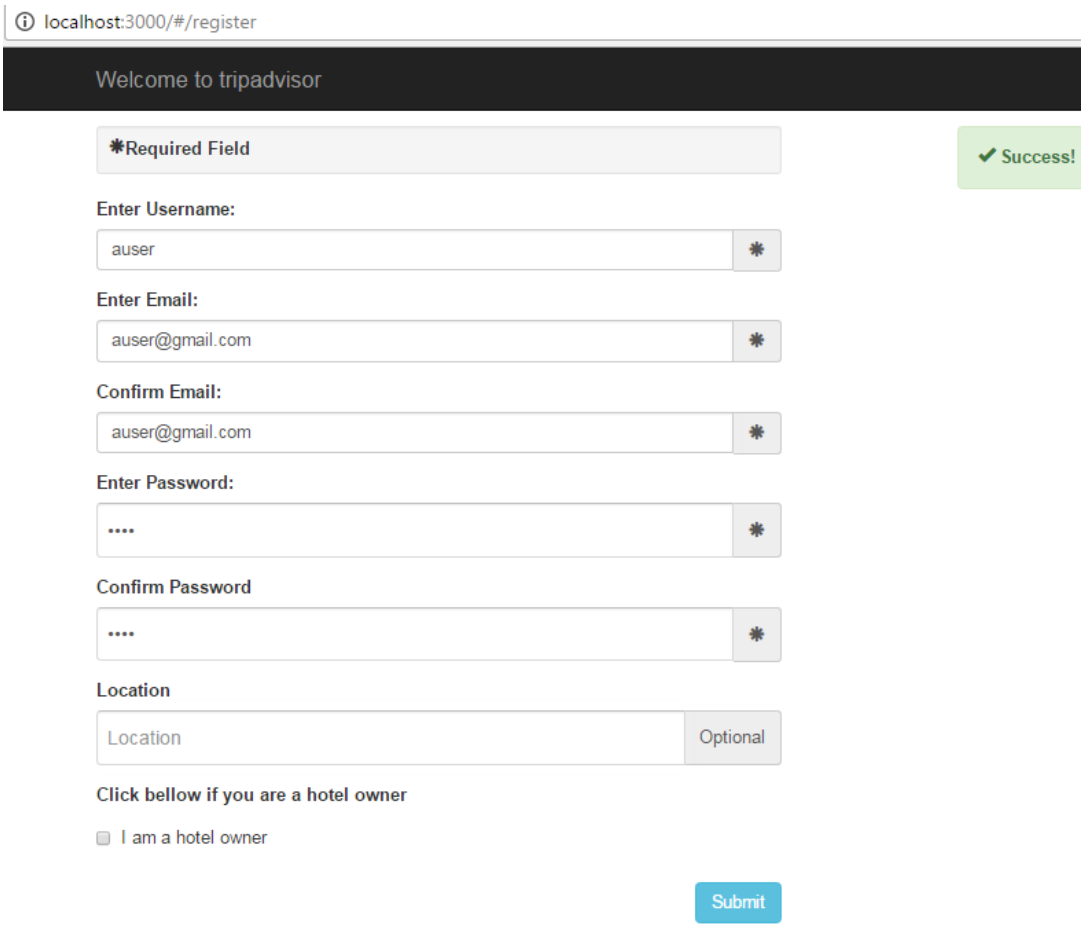
Πρόκειται για τη σελίδα όπου εμφανίζονται τα αποτελέσματα σε μορφή thumbnail όπως προκύπτουν από την αναζήτηση στην μπάρα αναζήτησης. Εδώ διακρίνεται το αποτέλεσμα αναζήτησης με τη χρήση της συμβολοσειράς “a”. Παρατηρούμε τον σύνδεσμο ότι είναι της μορφής hotels/filter/<query string>.

Προφανώς, αλλάζοντας το <query string>, παίρνουμε αποτελέσματα νέας αναζήτησης

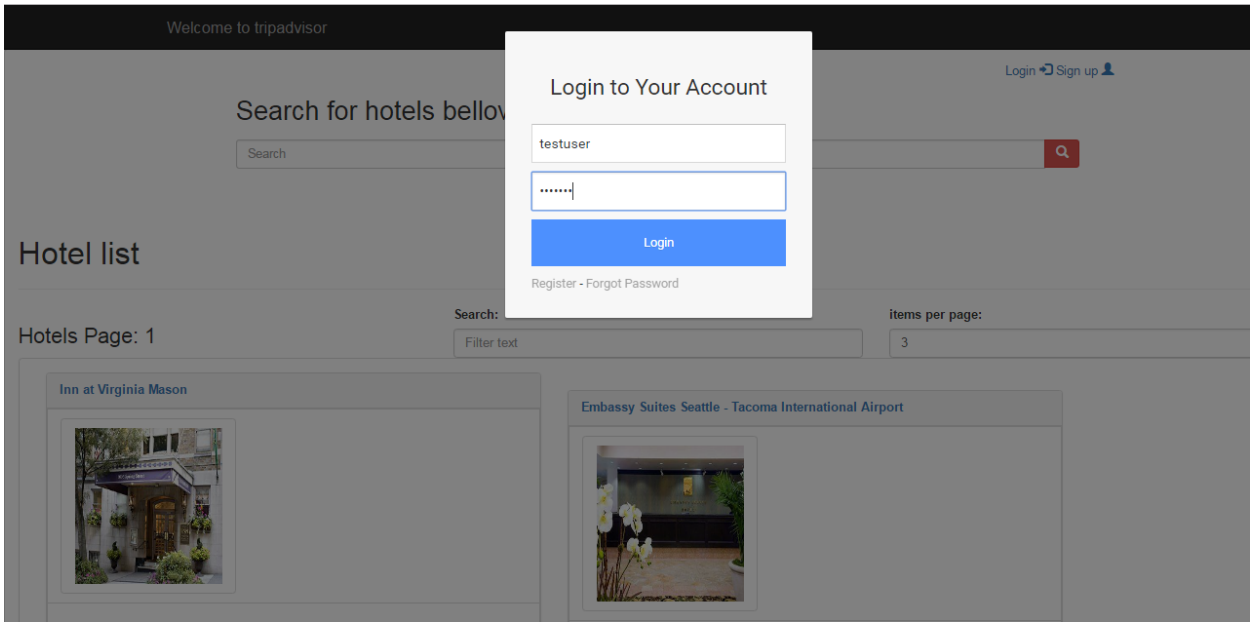


Εικόνα 23: Αποτέλεσμα επιλογής ξενοδοχείου και εμφάνιση κριτικών

Σε αυτό το σημείο, εμφανίζονται τα στοιχεία ενός ξενοδοχείου και οι κριτικές πάνω σε αυτό με τη μορφή αστεριών και ελεύθερου κειμένου. Παρατηρώντας και πάλι τον σύνδεσμο, διακρίνεται ότι είναι της μορφής hotels/<hotel id>.

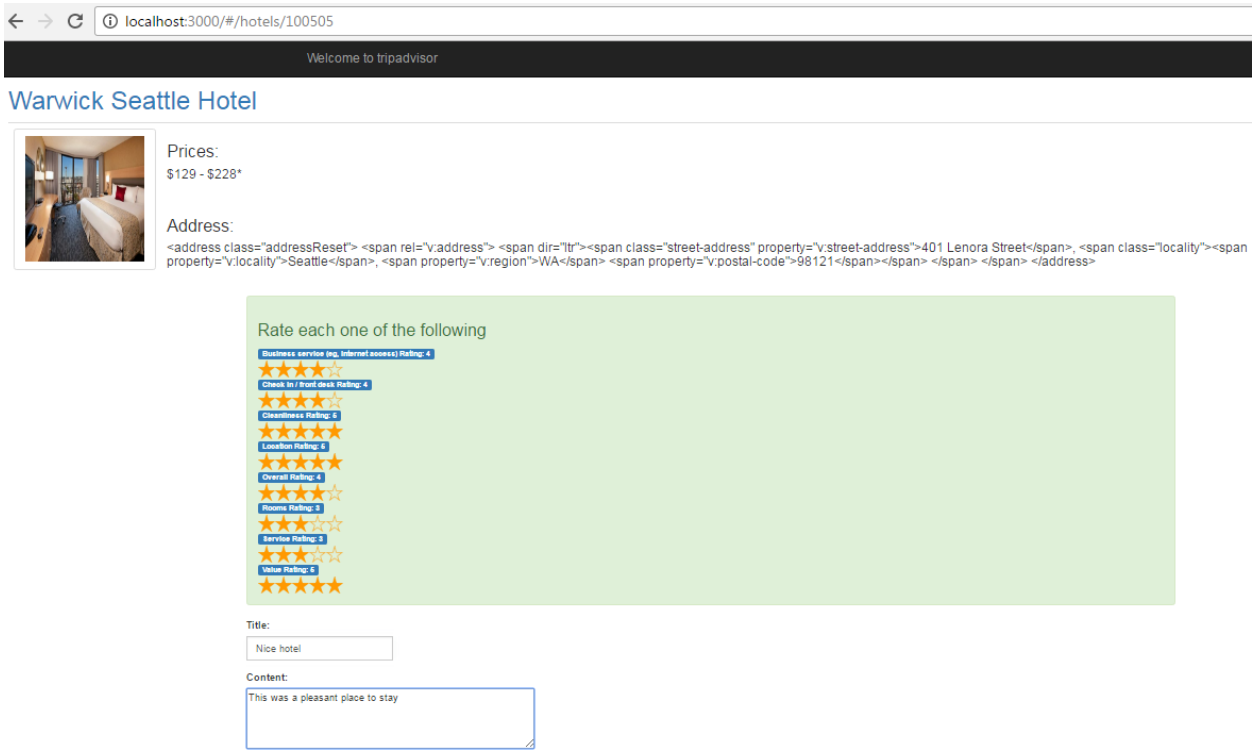


Εικόνα 24: Νέος Χρήστης



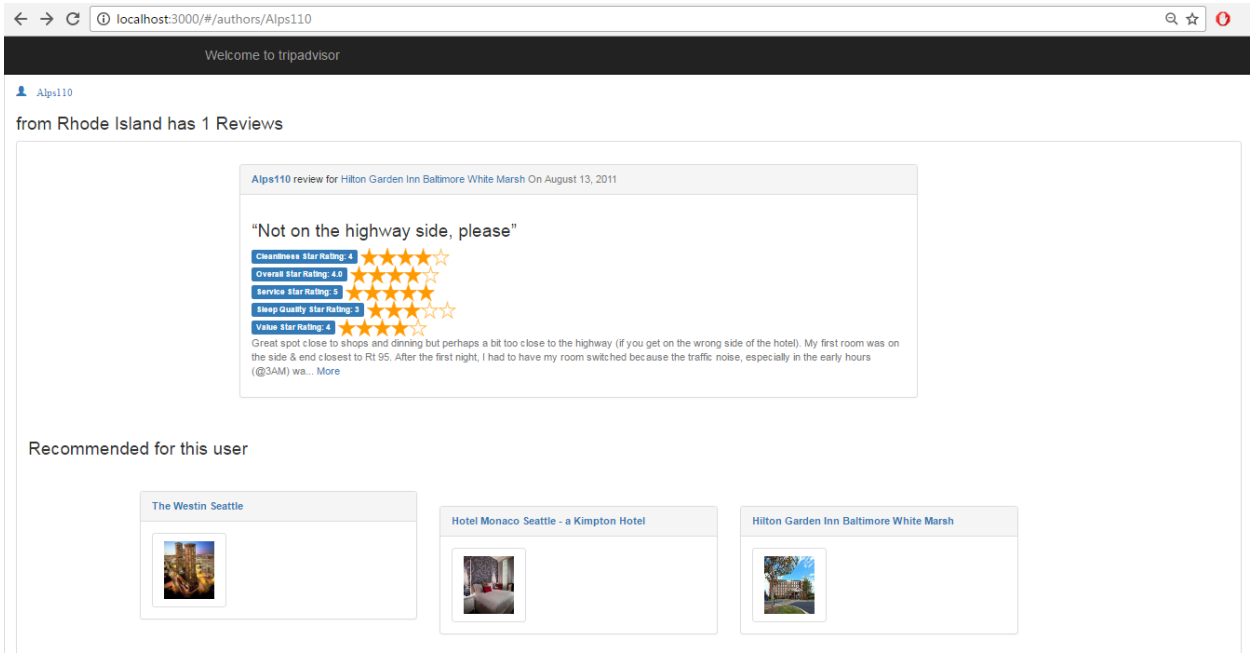
Εικόνα 25: Είσοδος Χρήστη στο σύστημα

Παρουσιάζεται η είσοδος εγγεγραμμένου χρήστη στο σύστημα. Ανεξάρτητα με την πλοήγηση που έχει προηγηθεί στο σύστημα, ένας χρήστης μπορεί να εισέλθει χρησιμοποιώντας τον σύνδεσμο Login και μετά την εισαγωγή του μέσω του modal παραθύρου που φαίνεται και στην εικόνα, συνεχίζει η πλοήγηση από το ίδιο σημείο.



Εικόνα 26: Εισαγωγή Νέας Κριτικής από Χρήστη

Σε αυτό το σημείο παρουσιάζεται η εισαγωγή μιας νέας κριτικής για το ξενοδοχείο. Ο χρήστης καλείται να εισάγει με τη μορφή αστεριών κριτική για κάθε ένα στοιχείο το ξενοδοχείου ξεχωριστά και στη συνέχεια του δίνεται η δυνατότητα μέσω ελεύθερου κειμένου να αναλύσει περεταίρω την κριτική του.



Εικόνα 27: Προφίλ Χρήστη

Τέλος, παρουσιάζεται η σελίδα χρήστη, όπου εμφανίζεται μία λίστα με όλες τις κριτικές που έχει πραγματοποιήσει ο συγκεκριμένος χρήστης καθώς και τα αποτελέσματα από τις προτάσεις που έχουν γίνει στον συγκεκριμένο χρήστη από το σύστημα παραγωγής προτάσεων.

Παρατηρούμε τον σύνδεσμο ότι είναι της μορφής authors/<author id>.

ΑΝΑΦΟΡΕΣ

[1] Big Data - What it is and why it matters http://www.sas.com/en_us/insights/big-data/what-is-big-data.html. [Προσπελάστηκε 12/11/2016]

[2] Viktor Mayer-Schönberger, Kenneth Cukier, *Big Data: A Revolution that Will Transform how We Live, Work, and Think*, Houghton Mifflin Harcourt, 2013.

[3] What is NoSQL, https://www.mongodb.com/nosql-explained. [Προσπελάστηκε 12/11/2016]

[4] Relational Vs Non Relational Database, https://www.mongodb.com/scale/relational-vs-non-relational-database. [Προσπελάστηκε 12/11/2016]

[5] ΓΚΟΤΣΕ ΜΠΛΕΡΙΝΑ, «Έξυπνες ροές εργασίας RESTful διαδικτυακών υπηρεσιών με τη βοήθεια του εργαλείου WebHookIt», Διπλωματική Εργασία, Τμήμα ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ, ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ, Ιούλιος 2015.

[6] Dr. M. Elkstein, *What is REST* . http://rest.elkstein.org/2008/02/what-is-rest.html. [Προσπελάστηκε 22/11/2016]

[7] Node.js, https://nodejs.org. [Προσπελάστηκε 29/11/2016]

[8] Strongloop, https://strongloop.com/. [Προσπελάστηκε 2/12/2016]

[9] What can you do with MongoDB, https://www.mongodb.com/what-is-mongodb. [Προσπελάστηκε 9/12/2016]

[10] Hadoop, http://hadoop.apache.org/. [Προσπελάστηκε 9/12/2016]

[11] Spark, http://spark.apache.org/. [Προσπελάστηκε 9/12/2016]

[12] Cloudera vm Images, http://www.cloudera.com/downloads.html. [Προσπελάστηκε 9/12/2016]

[13] Python, https://www.python.org/ . [Προσπελάστηκε 9/12/2016]

[14] Scala, https://www.scala-lang.org/. [Προσπελάστηκε 9/12/2016]

[15] Shiva Achari, *Hadoop Essentials*, Packt Publishing Ltd, 29 April 2015, pp. 33-37

[16] Facebook by the Numbers - Here's How Big It Really Is, https://www.thestreet.com/story/13352739/2/facebook-by-the-numbers-here-s-how-big-it-really-is.html. [Προσπελάστηκε 9/11/2016]

[17] MongoDB, https://www.mongodb.com/. [Προσπελάστηκε 9/11/2016]

[18] Node.js and the new web front-end, https://www.nczonline.net/blog/2013/10/07/node-js-and-the-new-web-front-end/. [Προσπελάστηκε 9/11/2016]

[19] Hadoop and MongoDB Use Cases, https://docs.mongodb.com/ecosystem/use-cases/hadoop/. [Προσπελάστηκε 16/10/2016]

[20] MongoDB and Hadoop: Driving Business Insights, pp 38, http://www.slideshare.net/mongodb/mongodb-and-hadoop-driving-business-insights. [Προσπελάστηκε 16/10/2016]

[21] What is a “Hadoop”? Explaining Big Data to the C-Suite, https://practicalanalytics.co/2011/11/06/explaining-hadoop-to-management-whats-the-big-data-deal/. [Προσπελάστηκε 16/10/2016]