ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη κατανεμημένου συστήματος σε υπολογιστικό νέφος

Αλέξανδρος Ευθυμιάδης
Ιωάννης Χαλδαίος

Εισηγητής: Δρ Παναγιώτης Γιαννακόπουλος, Καθηγητής

ΑΘΗΝΑ
ΙΟΥΝΙΟΣ 2018

(Κενό φύλλο)

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη κατανεμημένου συστήματος σε υπολογιστικό νέφος

Αλέξανδρος Ευθυμιάδης
Α.Μ. 40345

Ιωάννης Χαλδαίος
Α.Μ. 40977

Εισηγητής:

Δρ Παναγιώτης Γιαννακόπουλος, Καθηγητής

Εξεταστική Επιτροπή:

Δρ Παναγιώτης Πρεζεράκος, Καθηγητής
Δρ Δημήτρης Νικολόπουλος, Αναπληρωτής Καθηγητής

Ημερομηνία εξέτασης 19/06/2018

(Κενό φύλλο)

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΩΝ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Οι κάτωθι υπογεγραμμένοι : Αλέξανδρος Ευθυμιάδης του Ηλία και Ιωάννης Χαλδαίος του Ιωάννου, με αριθμό μητρώου 40345 και 40977 αντίστοιχα, φοιτητές του Τμήματος Μηχανικών Η/Υ Συστημάτων Τ.Ε. του Α.Ε.Ι. Πειραιά Τ.Τ. πριν αναλάβουμε την εκπόνηση της Πτυχιακής Εργασίας μας, δηλώνουμε ότι ενημερωθήκαμε για τα παρακάτω:

«Η Πτυχιακή Εργασία (Π.Ε.) αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο του συγγραφέα, όσο και του Ιδρύματος και θα πρέπει να έχει μοναδικό χαρακτήρα και πρωτότυπο περιεχόμενο.

Απαγορεύεται αυστηρά οποιοδήποτε κομμάτι κειμένου της να εμφανίζεται αυτούσιο ή μεταφρασμένο από κάποια άλλη δημοσιευμένη πηγή. Κάθε τέτοια πράξη αποτελεί προϊόν λογοκλοπής και εγείρει θέμα Ηθικής Τάξης για τα πνευματικά δικαιώματα του άλλου συγγραφέα. Αποκλειστικός υπεύθυνος είναι ο συγγραφέας της Π.Ε., ο οποίος φέρει και την ευθύνη των συνεπειών, ποινικών και άλλων, αυτής της πράξης.

Πέραν των όποιων ποινικών ευθυνών του συγγραφέα σε περίπτωση που το Ίδρυμα του έχει απονείμει Πτυχίο, αυτό ανακαλείται με απόφαση της Συνέλευσης του Τμήματος. Η Συνέλευση του Τμήματος με νέα απόφασης της, μετά από αίτηση του ενδιαφερόμενου, του αναθέτει εκ νέου την εκπόνηση της Π.Ε. με άλλο θέμα και διαφορετικό επιβλέποντα καθηγητή. Η εκπόνηση της εν λόγω Π.Ε. πρέπει να ολοκληρωθεί εντός τουλάχιστον ενός ημερολογιακού 6μήνου από την ημερομηνία ανάθεσης της. Κατά τα λοιπά εφαρμόζονται τα προβλεπόμενα στο άρθρο 18, παρ. 5 του ισχύοντος Εσωτερικού Κανονισμού.»

(Κενό φύλλο)

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα πτυχιακή εργασία ολοκληρώθηκε μετά από επίμονες προσπάθειες, σε ένα ενδιαφέρον γνωστικό αντικείμενο, όπως αυτό των κατανεμημένων συστημάτων. Την προσπάθειά μας αυτή υποστήριξε ο επιβλέπων καθηγητής μας, τον οποίο θα θέλαμε να ευχαριστήσουμε.

Ακόμα θα θέλαμε να ευχαριστήσουμε τον κ. Νικόλαο Τσελίκα για τις πολύτιμες συμβουλές του και το οικογενειακό μας περιβάλλον που ανέχτηκε και υποστήριξε την όλη διαδικασία υλοποίησης της πτυχιακής μας όλο αυτό το διάστημα.

(Κενό φύλλο)

ΠΕΡΙΛΗΨΗ

Με την συνεχώς αυξανόμενη εξέλιξη της τεχνολογίας, τα κατανεμημένα συστήματα γίνονται ολοένα και πιο διαδεδομένα, διότι με αυτή τη τεχνολογία, η επεξεργασία των δεδομένων κατανέμεται σε πολλούς υπολογιστές και δε περιορίζεται σε μία μόνο μηχανή. Πρόκειται για ένα τεράστιο και σύνθετο πεδίο έρευνας, στον τομέα της πληροφορικής.

Ο στόχος αυτής της εργασίας είναι η μελέτη, ο σχεδιασμός και η υλοποίηση ενός καταναμημένου συστήματος με τη βοήθεια της τεχνολογίας sharding, που μας παρέχει η μη-σχεσιακή βάση δεδομένων, με προσανατολισμό εγγράφων, MongoDB. Τέλος, γίνεται μία σύγκριση των αποτελεσμάτων από τη χρήση των διάφορων μεθόδων sharding, Ranged Based Sharding και Hashed Ranged Sharding, και ποιο είναι το “σωστό” shard κλειδί.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: Μηχανική λογισμικού, Κατανεμημένα Συστήματα
ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: κατανεμημένα συστήματα, μη-σχεσιακές βάσεις δεδομένων, mongodb, εικονικοποίηση, docker, java, Ταυτοχρονισμός Πολλαπλών Εκδόσεων

ABSTRACT

With the ever-growing technological expansion of the world, distributed systems are becoming more and more widespread because they’re trying to distribute the load of a system, to multiple computers instead of one. Distributed system are a vast and complex field of study in computer science.

The purpose of this thesis is the research, design and development of a distributed system with the help of the NoSQL, document-oriented database, MongoDB, by using the Sharding technology. Finally we are doing a comparison of the different methods of sharding that MongoDB offers, Ranged Based Sharding and Hashed Ranged Sharding and what is the best shard key to use.

Keywords: distributed systems, NoSQL, mongodb, virtualization, docker, java, MultiVersion Concurrency

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

array:	πίνακας
balancer:	ο εξισορροπητής προσπαθεί να επιτύχει την ίση διανομή των δεδομένων σε όλους τους κόμβους
broadcasting:	μετάδοση πληροφορίας σε όλους τους υπολογιστές του υποδικτύου
bytecode:	ενδιάμεσος κώδικας, κώδικας byte
chunk:	ένα τμήμα (chunk), αποτελείται από ένα υποσύνολο των θρυμματισμένων δεδομένων
client:	είναι ο πελάτης που κάνει αιτήματα στον εξυπηρετητή όταν χρειάζεται κάποια υπηρεσία
collection:	είναι το αντίστοιχο των πινάκων στις σχεσιακές βάσεις δεδομένων
compiled:	μεταγλωττισμένο (στο σύνολο εντολών του επεξεργαστή ή γλώσσα μηχανής)
complex data structure:	πολύπλοκη δομή δεδομένω
compound key:	είναι ένα σύνθετο κλειδί, που αποτελείται από πολλά πεδία των εγγράφων
concurrency:	ταυτοχρονισμός, η ιδιότητα ενός συστήματος, να εκτελεί ταυτόχρονα πολλές διεργασίες
configuration file:	αρχείο ρυθμίσεων
configuration management tool:	εργαλείο διαχείρισης διαμόρφωσης λογισμικού
containers:	είναι ένα στιγμιότυπο μιας ελαφριάς έκδοσης εικονικής μηχανής
controller:	ελεγκτής
cookie:	το cookie είναι πληροφορία που στέλνεται από τον διακομιστή μιας ιστοσελίδας στο πρόγραμμα περιήγησης (browser) το οποίο αποθηκεύεται στον σκληρό δίσκο του χρήστη
CPU:	η κεντρική μονάδα επεξεργασίας (Central Processing Unit) είναι το κεντρικό εξάρτημα που

	επεξεργάζεται δεδομένα σε έναν υπολογιστή
data structure:	δομή δεδομένων
datasets:	ένα σύνολο δεδομένων
dependencies:	εξαρτήσεις ενός συστήματος (οι βιβλιοθήκες που χρειάζεται το λογισμικό για να λειτουργήσει)
deployment:	εγκατάσταση λογισμικού
docker image:	μια εικόνα που περιέχει μία ελαφριά έκδοση εικονικής μηχανής
dockerfile:	αρχείο docker είναι η συνταγή που περιέχει όλα τα απαραίτητα βήματα που απαιτούνται για τη δημιουργία μιας εικόνας docker
document:	στη βάση δεδομένων MongDB, τα δεδομένα αποθηκεύονται ως έγγραφα, το αντίστοιχο των πεδίων πινάκων που έχουμε στις σχεσιακές βάσεις δεδομένων
document databases:	είναι NoSQL μοντέλο, στο οποίο ομαδοποιείται κάθε key με μία πολύπλοκη δομή δεδομένων γνωστή ως έγγραφο
downtime:	ο χρόνος κατα τον οποίο ένας ή περισσότεροι κόμβοι δεν είναι διαθέσιμοι σε ένα σύστημα, με συνέπεια, να μην είναι διαθέσιμες κάποιες υπηρεσίες / δεδομένα
e-mail:	ηλεκτρονικό ταχυδρομείο
graph databases:	είναι NoSQL μοντέλο, στο οποίο χρησιμοποιούνται δομές γράφων και είναι βασισμένες σε κόμβους (nodes), τις σχέσεις μεταξύ αυτών των κόμβων και τις ιδιότητές τους
hardware:	υλικό, ορίζεται το σύνολο των φυσικών εξαρτημάτων ενός υπολογιστή, όπως ηλεκτρικά και ηλεκτρονικά στοιχεία, μικροτσίπ
hash:	ο κατατεμαχισμός, είναι μια μαθηματική συνάρτηση που έχοντας ως είσοδο μια αυθαίρετου μεγέθους ομάδα δεδομένων δίνει

	έξοδο μια καθορισμένου μεγέθους στοιχειοσειρά
hashed Ranged Sharding:	το Hashed Ranged Sharding περιλαμβάνει τον υπολογισμό hash του πεδίου τιμής του θρυματισμένου κλειδιού. Κάθε τμήμα ανατίθεται ένα εύρος βασισμένο στις hashed τιμές του shard κλειδιού
high availability:	μεγάλη διαθεσιμότητα, η δυνατότητα μιας θρυμματισμένης συστοιχίας να μπορεί να συνεχίσει να εκτελεί εν μέρη λειτουργίες ανάγνωσης / εγγραφής ακόμη και αν δεν είναι διαθέσιμο ένα ή περισσότερα θραύσματα
highly available load balancer:	υψηλής διαθεσιμότητας εξισορρόπησης φόρτου
host:	Ο όρος Host αναφέρεται σε έναν υπολογιστή φιλοξενίας στον οποίο είναι αποθηκευμένη μια ιστοσελίδα ώστε να είναι προσβάσιμη από χρήστες μέσω Διαδικτύου.
high query rates:	υψηλά ποσοστά αιτημάτων
horizontal scaling / scale-out:	η οριζόντια επεκτασιμότητα περιλαμβάνει τη διαίρεση του συνόλου δεδομένων του συστήματος και τη διανομή αυτού του φόρτου σε πολλούς εξυπηρετητές
IDE:	ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated development environment) λογισμικού
immutable:	αμετάβλητο, που δεν μπορεί να αλλάξει τιμή
index:	είναι ένα κλειδί/δείκτης σε ένα πίνακα συσχέτισης
infrastructure:	υποδομή
insert:	είναι η μέθοδος της MongoDB με την οποία εισάγουμε νέα δεδομένα
interactions:	αλληλεπιδράσεις
jdbc:	ο οδηγός (Java DataBase Connectivity) ο οποίος παρέχει στην εφαρμογή επικοινωνία με τη βάση δεδομένων

json:	το JSON (JavaScript Object Notation) είναι ένα ελαφρύ πρότυπο ανταλλαγής δεδομένων
kernel:	πυρήνας λειτουργικού συστήματος
key-array:	ζευγάρι κλειδί-πίνακα, ένα από τους τρόπους αποθήκευσης δεδομένων στη MongoDB
key-value:	ζευγάρι κλειδί-τιμή, ένα από τους τρόπους αποθήκευσης δεδομένων στη MongoDB
key-value stores:	είναι NoSQL μοντέλο. Κάθε αντικείμενο της βάσης αποθηκεύεται ως ένα κλειδί (key) μαζί με τη τιμή του (value)
layer:	επίπεδο
learning curve:	καμπύλη δυσκολίας εκμάθησης
maven:	είναι εργαλείο διαχείρισης έργων λογισμικού
MegaByte:	είναι μονάδα μέτρησης ποσότητας πληροφορίας στα υπολογιστικά συστήματα
migration:	μετεγκατάσταση/μεταφορά δεδομένων από ένα κόμβο σε έναν άλλο
migration threshold:	όριο μεταγκατάστασης
multithreading:	η πολυνημάτωση αποτελεί ένα ευρέως διαδεδομένο μοντέλο προγραμματισμού και εκτέλεσης διεργασιών το οποίο επιτρέπει την ύπαρξη πολλών νημάτων μέσα στα πλαίσια μιας και μόνο διεργασίας
model:	μοντέλο
node:	ως κόμβος μπορεί να χαρακτηριστεί οποιαδήποτε ενεργή ηλεκτρονική συσκευή που μπορεί να επικοινωνήσει με άλλες συσκευές σε ένα συνδεδεμένο δίκτυο τοπικά ή μέσω διαδικτύου
non-sharded collections:	μη-θρυμματισμένες συλλογές, είναι οι συλλογές που δεν έχουν θρυμματιστεί
object oriented:	αντικειμενοστρεφής, χαρακτηρίζεται μια γλώσσα προγραμματισμού, όταν ο χειρισμός σχετιζόμενων δεδομένων και των διαδικασιών που επενεργούν

	σε αυτά γίνεται από κοινού, μέσω μίας δομής δεδομένων που τα περιβάλλει ως αυτόνομη οντότητα με ταυτότητα και δικά της χαρακτηριστικά
open source:	είναι ένα λογισμικό το οποίο διατίθεται με ειδικές άδειες, οι οποίες επιτρέπουν στους χρήστες να μελετήσουν, να τροποποιήσουν και να βελτιώσουν το λογισμικό
pointer:	δείκτης, δείκτης διεύθυνσης
POM:	μοντέλου αντικειμένου έργου (Project Object Model), το οποίο διαχειρίζεται το χτίσιμο και της εξαρτήσεις ενός έργου
project:	έργο
queries:	είναι τα ερωτήματα προς μια βάση δεδομένων
proxy server:	διακομιστής μεσολάβησης
ranged Based Sharding:	Το Ranged Based Sharding περιλαμβάνει τη διαίρεση των δεδομένων σε φάσματα βασισμένο στο θρυμματισμένο κλειδί. Έπειτα σε κάθε τμήμα ανατίθεται ένα εύρος βασισμένο στις τιμές του shard κλειδιού
read:	ανάγνωση
redirect:	η τεχνική της ανακατεύθυνσης ή redirection κάνει μια ιστοσελίδα να είναι διαθέσιμη σε περισσότερες από μια διευθύνσεις URL
redundancy:	η εφεδρεία είναι η ικανότητα ενός συστήματος να παρέχει τις υπηρεσίες του, ακόμα και όταν κάποιοι από τους πόρους του συστήματος δεν είναι διαθέσιμοι
relational database management system:	σχεσιακές βάσεις δεδομένων σύστημα διαχείρισης
resource sharing:	ο διαμοιρασμός των πόρων του δικτύου, δηλαδή η ταυτόχρονη χρήση τους από πολλούς χρήστες/εφαρμογές

script:	κομμάτι κώδικα
server:	ένας διακομιστής/εξυπηρετητής είναι ένας υπολογιστής που παρέχει διάφορες υπηρεσίες και αναλαμβάνει να εξυπηρετήσει τα αιτήματα άλλων υπολογιστών ή εφαρμογών γνωστών ως πελάτες
service:	υπηρεσία
session:	περιγράφει ένα σύνολο αλληλεπιδράσεων (interactions) του χρήστη που πραγματοποιούνται στην ιστοσελίδα/εφαρμογή μας, μέσα σε ένα δεδομένο χρονικό διάστημα
shard:	κάθε θραύσμα (shard) περιέχει ένα υποσύνολο των θρυμματισμένα (sharded) δεδομένων
shards:	θραύσματα
sharded cluster:	η συστοιχία θρυμμάτων, αποτελείται από πολλούς κόμβους στους οποίους είναι αποθηκευμένα τα θρυμματισμένα δεδομένα
sharding:	ο θρυμματισμός (sharding) είναι μια μέθοδος για τη διανομή δεδομένων μεταξύ πολλών μηχανών (συστοιχίας υπολογιστών)
software project management tool:	εργαλείο διαχείρισης έργων λογισμικού εργαλείο διαχείρισης
SSH:	Το SSH (Secure Shell) είναι ένα ασφαλές δικτυακό πρωτόκολλο το οποίο επιτρέπει τη μεταφορά δεδομένων μεταξύ δύο υπολογιστών.
structures και unions:	δομές και ενώσεις
openness:	ευρύτητα, χαρακτηρίζει τη δυνατότητα ενός συστήματος να παρέχει τμήματα του σε άλλες εφαρμογές/συστήματα
RAM:	Μνήμη Τυχαίας Προσπέλασης, είναι όρος που χρησιμοποιούμε για ηλεκτρονικές διατάξεις προσωρινής αποθήκευσης ψηφιακών δεδομένων (μνήμης υπολογιστή), οι οποίες επιτρέπουν πρόσβαση στα αποθηκευμένα δεδομένα στον ίδιο

	χρόνο οπουδήποτε και αν βρίσκονται αυτά, δηλαδή με «τυχαία πρόσβαση»
scalability:	κλιμακωσιμότητα, η δυνατότητα αύξησης της απόδοσης με την προσθήκη υλικού
schema less:	είναι ένα από τα χαρακτηριστικά της MongoDB. Ο αριθμός των πεδίων, το περιεχόμενο και το μέγεθος των εγγράφων μπορούν να διαφέρουν από το ένα έγγραφο στο άλλο
single entity:	μία οντότητα
storage capacity:	χωρητικότητα αποθήκευσης
table:	πίνακας
template engine:	μηχανή προτύπου
templates:	πρότυπα
text files:	αρχεία κειμένου
transparency:	διαφάνεια, η προσπάθεια ενός συστήματος να κάνει τη διανομή των διαδικασιών και πόρων, διαφανής, δηλαδή αόρατη στους τελικούς χρήστες και τις εφαρμογές
unique identifier field:	μοναδικό αναγνωριστικό πεδίο, χρησιμοποιείται για να μπορούμε να ξεχωρίσουμε τα έγγραφα της MongoDB
update:	είναι η μέθοδος της MongoDB με την οποία τροποποιούμε υπάρχοντα δεδομένα
URL:	Ένα Url (Uniform Resource Locator) είναι ένας τύπος URI που χρησιμοποιείται για να προσδιορίσει την διεύθυνση ενός πόρου στο Διαδίκτυο ο οποίος αναφέρεται πολλές φορές και ως σύνδεσμος
user data:	δεδομένα χρηστών
vertical scaling / scale-up:	η Κατακόρυφη Επεκτασιμότητα συνεπάγεται την αύξηση της χωρητικότητας ενός μόνο εξυπηρετητή, όπως η χρήση πιο ισχυρής CPU, η προσθήκη περισσότερης μνήμης RAM ή η αύξηση

	του χώρου αποθήκευσης
view:	προβολή
virtual machine	εικονική μηχανή, είναι μια εξομοίωση ενός ξεχωριστού υπολογιστή μέσα στον υπολογιστή μας
web:	διαδίκτυο
wide-column stores:	είναι NoSQL μοντέλο, στο οποίο τα δεδομένα αποθηκεύονται σε κελιά (cells), τα οποία ομαδοποιούνται σε στήλες (columns) και όχι σε σειρές (rows) και είναι βελτιστοποιημένα για να επεξεργάζονται αιτήματα με μεγάλο όγκο δεδομένων
write:	εγγραφή

(Κενό φύλλο)

ΚΕΦΑΛΑΙΟ 1

1. ΕΙΣΑΓΩΓΗ

Σε αυτό το κεφάλαιο αναλύεται το αντικείμενο της πτυχιακής εργασίας, τα κατανεμημένα συστήματα.

1.1 Κατανεμημένα Συστήματα

Με την ολοένα αυξανόμενη τεχνολογική ανάπτυξη, τα κατανεμημένα συστήματα γίνονται ολοένα και πιο διαδεδομένα. Πρόκειται για ένα τεράστιο και σύνθετο πεδίο έρευνας στον τομέα της πληροφορικής.

Ένα κατανεμημένο σύστημα αποτελείται από πολλαπλά τμήματα λογισμικού που βρίσκονται σε πολλαπλούς υπολογιστές, αλλά λειτουργούν ως ένα ενιαίο σύστημα. Οι υπολογιστές που βρίσκονται σε ένα κατανεμημένο σύστημα μπορούν να βρίσκονται αρκετά κοντά και να να είναι συνδεδεμένοι μέσω ενός τοπικού δικτύου (local network) ή να είναι γεωγραφικά απομακρυσμένοι και να είναι συνδεδεμένοι μέσω ενός δικτύου ευρείας περιοχής (wide area network).

Τα κατανεμημένα συστήματα χαρακτηρίζονται από διαμοιρασμό πόρων (resource sharing), ταυτοχρονισμό (concurrency), ευρύτητα (openness) κλιμακωσιμότητα (scalability), εφεδρεία (redundancy), διαφάνεια (transparency). [14]

1.1.1 Διαμοιρασμός πόρων

Μία από τις σημαντικότερες ιδιότητες ενός κατανεμημένου συστήματος είναι να διευκολύνει τους χρήστες (και τις εφαρμογές) να έχουν πρόσβαση και να μοιράζονται απομακρυσμένους πόρους. Οι πόροι μπορεί να είναι σχεδόν οτιδήποτε, τυπικά παραδείγματα περιλαμβάνουν περιφερειακά, μονάδες δίσκων προς αποθήκευση, δεδομένα, αρχεία, υπηρεσίες και δίκτυα, είναι μερικά από αυτά. Υπάρχουν πολλοί λόγοι για τους οποίους επιθυμούμε να μοιραστούμε πόρους. Ένας προφανής λόγος είναι αυτός των χρημάτων. Για παράδειγμα, είναι

φθηνότερο να υπάρχει μια κοινόχρηστη μονάδα αποθήκευσης υψηλού επιπέδου, ο οποίος θα πρέπει να μοιράζεται, σε αντίθεση με το να έχουμε μία ξεχωριστή μονάδα αποθήκευσης για το κάθε χρήστη.

1.1.2 Ταυτοχρονισμός

Είναι η ιδιότητα ενός συστήματος, να εκτελεί ταυτόχρονα πολλές διεργασίες, που μπορούν να αλληλεπιδρούν μεταξύ τους. Οι διεργασίες μπορούν να εκτελούνται σε έναν ή σε πολλούς υπολογιστές παράλληλα.

1.1.3 Ευρύτητα

Ένας άλλος σημαντικός στόχος των κατανεμημένων συστημάτων είναι η ευρύτητα. Η ευρύτητα είναι ουσιαστικά η ιδιότητα του συστήματος να μπορεί να αξιοποήσει υλικό (hardware) από διάφορους κατασκευαστές, και λογισμικό από άλλους προγραμματιστές/εταιρείες, ενώ παράλληλα να μπορεί να προσφέρει τμήματα του, προς χρησιμοποίηση/ενσωμάτωση σε άλλα κατανεμημένα συστήματα.

1.1.4 Κλιμακωσιμότητα (Scalability)

Η κλιμακωσιμότητα/επεκτασιμότητα είναι ίσως η σημαντικότερη μέθοδος αξιολόγησης των σύγχρονων συστημάτων παράλληλης επεξεργασίας. Εκφράζει την ολοένα αυξανόμενη ζήτηση για μεγαλύτερη απόδοση του συστήματος, καθώς προσθέτουμε ολοένα και περισσότερους πόρους σε αυτό. Γενικά υπάρχουν δύο μορφές επεκτασιμότητας : η οριζόντια (Horizontal Scaling/Scale Out) που σημαίνει συνήθως αύξηση του αριθμού των πόρων (π.χ. προσθήκη περισσότερων κόμβων σε μια συστοιχία υπολογιστών) και η κατακόρυφη (Vertical Scaling/Scale Up) που σημαίνει βελτίωση των ήδη υπαρχόντων πόρων (με ισχυρότερες CPUs ή περισσότερη και γρηγορότερη μνήμη RAM).

1.1.4.1 Κατακόρυφη Επεκτασιμότητα

Η Κατακόρυφη Επεκτασιμότητα συνεπάγεται την αύξηση της χωρητικότητας ενός μόνο εξυπηρετητή, όπως η χρήση πιο ισχυρής CPU, η προσθήκη περισσότερης μνήμης RAM ή η αύξηση του χώρου αποθήκευσης. Ο περιορισμός της συγκεκριμένης τεχνολογίας είναι ότι το ένα και μοναδικό μηχάνημα που έχουμε στη διάθεσή μας, να μην είναι αρκετά ισχυρό για ένα συγκεκριμένο φόρτο εργασίας. Επιπλέον, οι Παροχείς Τεχνολογίας Σύννεφου (Cloud-based providers) έχουν σκληρά ανώτατα όρια βάσει των διαθέσιμων διαμορφώσεων υλικού (hardware configurations). Ως αποτέλεσμα, υπάρχει ένα μέγιστο όριο για τη κατακόρυφη κλιμάκωση.

1.1.4.2 Οριζόντια Επεκτασιμότητα

Η Οριζόντια Επεκτασιμότητα περιλαμβάνει τη διαίρεση του συνόλου δεδομένων του συστήματος και τη διανομή αυτού του φόρτου σε πολλούς εξυπηρετητές, προσθέτοντας επιπλέον εξυπηρετητές για να αυξήσετε την απαιτούμενη χωρητικότητα/απόδοση του συστήματος. Παρόλο που η συνολική ταχύτητα ή η χωρητικότητα ενός μηχανήματος μπορεί να μην είναι αρκετά υψηλή, κάθε μηχάνημα χειρίζεται ένα υποσύνολο του συνολικού φόρτου εργασίας, ενδεχομένως παρέχοντας καλύτερη απόδοση από έναν μεμονωμένο εξυπηρετητή μεγάλης ταχύτητας και υψηλής χωρητικότητας. Για την αύξηση της απόδοσης και της χωρητικότητας του συστήματος, το μόνο που χρειάζεται είναι η προσθήκη περισσότερων εξυπηρετητών, το οποίο μπορεί να κοστίσει λιγότερο από την αγορά ενός ισχυρού μηχανήματος μεγάλης απόδοσης και υψηλής χωρητικότητας. [6]

1.1.5 Εφεδρεία

Η εφεδρεία περιγράφει την ικανότητα ενός συστήματος να παρέχει τις υπηρεσίες του, ακόμα και όταν κάποιοι από τους πόρους του συστήματος δεν είναι διαθέσιμοι. Αυτό επιτυγχάνεται με το να έχουμε πόρους που μπορούν να παρέχουν τις ίδιες υπηρεσίες, οπότε αν δεν είναι διαθέσιμος κάποιος πόρος, θα αναλάβουν οι υπόλοιποι διαθέσιμοι πόροι και η εργασία δεν σταματά. Επιπλέον, επειδή μπορούν να χρησιμοποιηθούν πολλές μικρότερες μηχανές, αυτή η εφεδρεία δεν χρειάζεται να είναι απαγορευτικά δαπανηρή.

1.1.6 Διαφάνεια

Ένας από τους στόχους ενός κατανεμημένου συστήματος είναι να κρύψει το γεγονός ότι οι διαδικασίες του και οι πόροι διανέμονται φυσικά σε πολλούς υπολογιστές, που ενδεχομένως είναι χωρισμένοι σε μεγάλες αποστάσεις. Με άλλα λόγια, προσπαθεί να κάνει τη διανομή των διαδικασιών και πόρων, διαφανής, δηλαδή αόρατη στους τελικούς χρήστες και τις εφαρμογές, ώστε το σύστημα να φαίνεται ως ένας υπολογιστής. [13] [14]

1.2 Περιγραφή του αντικειμένου της πτυχιακής εργασίας

Ο στόχος αυτής της εργασίας είναι η μελέτη, ο σχεδιασμός και η υλοποίηση ενός καταναμημένου συστήματος με τη βοήθεια της τεχνολογίας sharding, που μας παρέχει η μη-σχεσιακή βάση δεδομένων, με προσανατολισμό εγγράφων, MongoDB. Τέλος, γίνεται μία σύγκριση των αποτελεσμάτων από τη χρήση των διάφορων μεθόδων sharding, Ranged Based Sharding και Hashed Ranged Sharding.

ΚΕΦΑΛΑΙΟ 2

2. ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

Στην παρούσα ενότητα περιγράφονται σημαντικές έννοιες και τεχνολογίες που απαιτούνται για την κατανόηση της εργασίας. Συγκεκριμένα αναφέρονται οι μη-σχεσιακές βάσεις δεδομένων, τα κατανεμημένα συστήματα και η βάση δεδομένων MongoDB, στην οποία βασίζεται το σύστημά μας.

2.1 NoSQL βάσεις δεδομένων

2.1.1 Τι είναι οι NoSQL βάσεις δεδομένων

Στη σημερινή εποχή οι σχεσιακές βάσεις δεδομένων (RDBMS - Relational Database Management System) χρησιμοποιούνται στην συντριπτική πλειοψηφία των περιπτώσεων για την οργάνωση και αποθήκευση δεδομένων σε επαγγελματικές εφαρμογές και στο διαδίκτυο. Το RDBMS μοντέλο προσφέρει αρκετές δυνατότητες και ακριβή συνέπεια στα δεδομένα αλλά, δεν αποτελεί τη λύση για όλα τα προβλήματα καθώς πολλές εφαρμογές χρειάζονται περισσότερη ελευθερία, όπως επίσης και μεγαλύτερη απόδοση από αύτη που τους προσφέρει μια σχεσιακή βάση δεδομένων, και αν μη τι άλλο, εισάγει περιορισμούς στο horizontal scalability. Κατα συνέπεια αυτό μας οδήγησε στη δημιουργία νέων συστήματων βάσεων δεδομένων τα οποία είναι γνωστά σαν NoSQL βάσεις δεδομένων.

Ο όρος NoSQL όπως τον ξέρουμε σήμερα, προήλθε από την φράση No SQL (όχι SQL) την οποία εισήγαγε ο προγραμματιστής Johan Oskarsson στις 11 Ιουνίου 2009 σε μια συνάντηση στο Σαν Φρανσίσκο, κάτα την οποία παρουσιάστηκαν κάποιες «νέες» βάσεις δεδομένων. Σαφής ορισμός σχετικά με το τι είναι ένα σύστημα NoSQL δεν υπάρχει. Αναφέρεται σε μια ευρεία ομάδα συστημάτων διαχείρισης βάσεων δεδομένων (database management system) που το κύρια χαρακτηριστικά τους είναι [1]:

- δεν τηρούν το κλασικό σχεσιακό μοντέλο
- λειτουργούν αποδοτικά σε συστοιχία υπολογιστών (computer cluster)
- είναι ΕΛ/ΛΑΚ - Ελεύθερο Λογισμικό / Λογισμικό Ανοικτού Κώδικα (Open-Source)
- είναι σχεδιασμένες για τις ανάγκες του διαδικτύου του 21ου αιώνα
- Schemaless, δεν απαιτείται ένα ορισμένο schema στη δομή των αρχείων

2.1.2 Κατηγορίες NoSQL συστημάτων

Τα NoSQL συστήματα μπορούν να κατηγοριοποιηθούν ως εξής [2]:

Key-value stores είναι το πιο σχετικά απλό και εύκολο στην υλοποίηση μοντέλο από τις NoSQL βάσεις. Κάθε αντικείμενο της βάσης αποθηκεύεται ως ένα κλειδί (key) μαζί με τη τιμή του (value). Riak, Voldemort και Redis είναι οι πιο γνωστές βάσεις δεδομένων αυτής της κατηγορίας.

Wide-column stores αποθηκεύουν τα δεδομένα σε κελιά (cells), τα οποία ομαδοποιούνται σε στήλες (columns) και όχι σε σειρές (rows) και είναι βελτιστοποιημένα για να επεξεργάζονται queries σε μεγάλα datasets. Οι πιο δημοφιλείς είναι η Cassandra και Hbase.

Document databases ομαδοποιούν κάθε key με μία πολύπλοκη δομή δεδομένων (complex data structure) γνωστή ως έγγραφο (document). Τα έγγραφα (documents) μπορούν να περιέχουν πολλά διαφορετικά ζευγάρια key-value, ή key-array, είτε ακόμη και εμφωλευμένα έγραφα (nested documents). Η MongoDB είναι η πιο δημοφιλής βάση δεδομένων αυτής της κατηγορίας.

Graph databases χρησιμοποιούν δομές γράφων και είναι βασισμένες σε κόμβους (nodes), τις σχέσεις μεταξύ αυτών των κόμβων και τις ιδιότητές τους. Αντί για πίνακες με στήλες και σειρές, εδώ υπάρχει ένα ευέλικτο γραφικό μοντέλο (graphmodel) που μπορεί να χρησιμοποιηθεί και να αναπτυχθεί παράλληλα σε

πολλά μηχανήματα (servers – κόμβους). Τέτοια παραδείγματα είναι οι Neo4J και HyperGraphDB.

2.2 MongoDB

2.2.1 Εισαγωγή

Η MongoDB είναι μία μη σχεσιακή (NoSQL) βάση δεδομένων ανοιχτού κώδικα και συνεπώς ελεύθερης διάθεσης. Πρόκειται, ίσως για τη δημοφιλέστερη βάση δεδομένων με προσανατολισμό εγγράφων (document-oriented) που υπάρχει αυτή τη στιγμή. Η λέξη Mongo, προέρχεται από τη λέξη humongous που σημαίνει τεράστιος. Περιγράφεται ως NoSQL, διότι δε χρησιμοποιεί σχεσιακή δομή, βασιζόμενης σε πίνακες, αλλά αποθηκεύει τα δεδομένα ως documents μέσα σε μια συλλογή (collection). Τα documents αποθηκεύονται ως BSON (Binary JSON), μια συγγενική μορφή της JSON (JavaScript Object Notation). Με αυτό το τρόπο προσφέρεται ευκολία και ταχύτητα στην εισαγωγή των δεδομένων σε συγκεκριμένους τύπους εφαρμογών. [3]

2.2.2 Πλεονεκτήματα

Μερικά από τα πλεονεκτήματα της MongoDB είναι:

- Schema less − Η MongoDB είναι μια βάση δεδομένων με προσανατολισμό εγγράφων στην οποία ένα collection περιέχει διαφορετικά documents. Ο αριθμός των πεδίων, το περιεχόμενο και το μέγεθος του document μπορούν να διαφέρουν από το ένα document στο άλλο.
- Από τη στιγμή που η MongoDB είναι document-based βάση δεδομένων, έχει μια πλούσια Δομή Δεδομένων (Data Structure) η οποία είναι ικανή να κρατά πίνακες (arrays) και άλλα documents. Το οποίο σημαίνει ότι πολλές φορές μπορείς να αναπαραστήσεις σε μία οντότητα (single entity) μία δομή δεδομένων, η οποία θα απαιτούσε αρκετούς πίνακες (tables) ώστε να αναπαρασταθεί σωστά σε μια Σχεσιακή Βάση Δεδομένων. Αυτό είναι ιδιαίτερα χρήσιμο εάν τα δεδομένα σας είναι αμετάβλητα (immutable).

- Υποστηρίζει μία αρκετά πλούσια γλώσσα ερωτημάτων, η οποία είναι εξίσου ισχυρή με αυτήν της SQL.
- Απουσία πολύπλοκων ενώσεων (joins) πινάκων.
- Υποστηρίζει Οριζόντια Επεκτασιμότητα (Horizontal Scaling/Scale Out) που σημαίνει συνήθως αύξηση του αριθμού των πόρων (π.χ. προσθήκη περισσότερων κόμβων σε μια συστοιχία υπολογιστών), μέσω της τεχνολογίας θρυμματισμού (sharding) [4]

2.2.3 Mongo Java Driver

Είναι ο οδηγός (driver) JDBC (**J**ava **D**ata**B**ase **C**onnectivity) ο οποίος παρέχει στην εφαρμογή (application) σύγχρονη (synchronous) και ασύγχρονη (asynchronous) επικοινωνία με τη Βάση Δεδομένων MongoDB. [5]

2.2.4 Sharding

2.2.4.1 Περιγραφή του Sharding

Ο Θρυμματισμός (Sharding) είναι μια μέθοδος για τη διανομή δεδομένων μεταξύ πολλών μηχανών (συστοιχίας υπολογιστών). Η MongoDB χρησιμοποιεί sharding για να υποστηρίξει deployments με πολύ μεγάλα σύνολα δεδομένων και εφαρμογές υψηλής απόδοσης.

Τα συστήματα βάσεων δεδομένων με μεγάλα σύνολα δεδομένων ή εφαρμογές υψηλής απόδοσης μπορούν να αμφισβητήσουν την ικανότητα ενός μόνο εξυπηρετητή (single server). Για παράδειγμα, υψηλά ποσοστά αιτημάτων (high query rates) μπορούν να εξαντλήσουν τις δυνατότητες του επεξεργαστή (CPU - Central Processing Unit) ενός εξυπηρετητή, είτε να υπάρχει ζήτηση για περισσότερη Μνήμη Τυχαίας Προσπέλασης (RAM - Random Access Memory) από αυτή που παρέχει το σύστημα, δημιουργόντας μεγάλο φόρτο στη μονάδα δίσκου.

Υπάρχουν δύο μέθοδοι για την αντιμετώπιση της ολοένα αυξανόμενης ζήτησης για μεγαλύτερη απόδοση του συστήματος, καθώς προσθέτουμε ολοένα και περισσότερους πόρους σε αυτό, με κατακόρυφη (Vertical Scaling/Scale Up) και οριζόντια επεκτασιμότητα (Horizontal Scaling/Scale Out) όπως περιγράφθηκαν στην ενότητα 1.1.4 Κλιμακωσιμότητα.

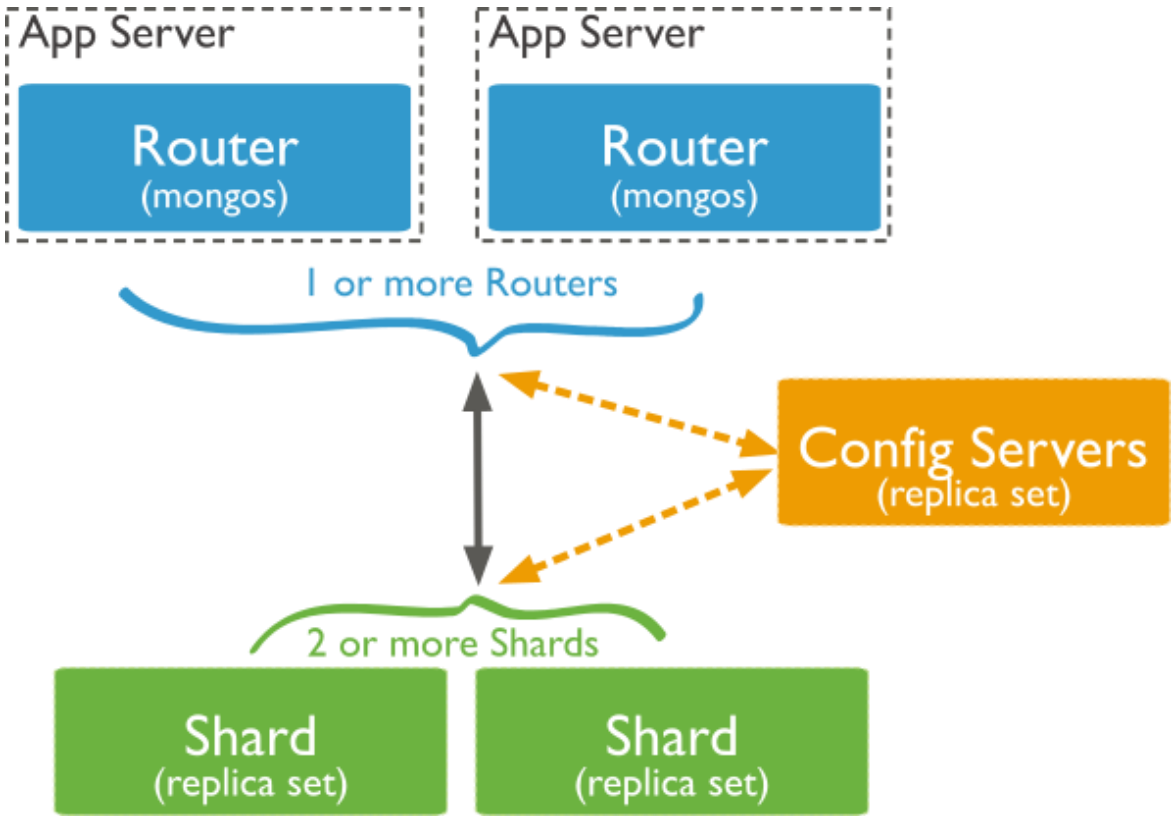
Η MongoDB υποστηρίζει Οριζόντια Επεκτασιμότητα μέσω της τεχνολογίας Sharding.

2.2.4.2 Sharded Cluster

Μία MongoDB Συστοιχία Θρυμμάτων (Sharded Cluster) αποτελείται από τα ακόλουθα στοιχεία:

- shard: Κάθε θραύσμα (shard) περιέχει ένα υποσύνολο των θρυμματισμένα (sharded) δεδομένων. Κάθε shard μπορεί να αναπτυχθεί ως σύνολο αντιγράφων (replica set).
    - replica set: Είναι ένα σύμπλεγμα MongoDB εξυπηρετητών που υλοποιεί την δημιουργία αντιγράφου (replication) του μοντέλου master-slave και διαχειρίζεται αυτόματα την αποτυχία του συστήματος (automated failover).
    - replication: Μια δυνατότητα που επιτρέπει σε πολλούς εξυπηρετητές βάσεων δεδομένων να μοιράζονται τα ίδια δεδομένα, εξασφαλίζοντας έτσι την εφεδρεία των δεδομένων και διευκολύνοντας την εξισορρόπηση του φόρτου του συστήματος (load balancing).
- mongos: Το mongos λειτουργεί ως δρομολογητής ερωτήσεων (query router), παρέχοντας μια διεπαφή μεταξύ των εφαρμογών πελάτη (client applications) και του sharded cluster.
- config servers: Οι Config Εξυπηρετητές (Config Servers) αποθηκεύουν τα μεταδεδομένα (metadata) και τις ρυθμίσεις διαμόρφωσης (configuration settings) για τη συστοιχία υπολογιστών (cluster). Από το MongoDB 3.2, οι config servers μπορούν να αναπτυχθούν ως σύνολο αντιγράφων (replica set).

Στο ακόλουθο σχήμα (Σχήμα 2.1) περιγράφεται η αλληλεπίδραση των στοιχείων μέσα σε μία Συστοιχία Θρυμμάτων:



Σχήμα 2.1 Συστοιχία Θρυμμάτων

Η MongoDB θρυμματίζει (shards) τα δεδομένα στο επίπεδο συλλογής (collection level), διανέμοντας τα δεδομένα συλλογής (collection data) μεταξύ των θραυσμάτων (shards) της συστοιχίας υπολογιστών (cluster).

2.2.4.3 Shard Keys

Για να διανείμει τα έγγραφα (documents) σε μια συλλογή (collection), η MongoDB χωρίζει τη συλλογή χρησιμοποιώντας το shard κλειδί (shard key). Το shard κλειδί αποτελείται από ένα αμετάβλητο (immutable) πεδίο ή πεδία που υπάρχουν σε κάθε έγγραφο της συλλογής.

Επιλέγετε το shard κλειδί όταν θρυμματίζεται (sharding) μια συλλογή (collection). Η επιλογή του shard κλειδιού δεν μπορεί να αλλάξει μετά το sharding.

Μια θρυμματισμένη συλλογή (sharded collection) μπορεί να έχει μόνο ένα shard κλειδί.

Για να κάνετε shard μια μη κενή συλλογή (collection), η συλλογή πρέπει να έχει ένα ευρετήριο (index) που ξεκινά με το Shard κλειδί. Για κενές συλλογές, η MongoDB δημιουργεί το index αν η συλλογή δεν διαθέτει ήδη κατάλληλο index για το καθορισμένο shard κλειδί.

Η επιλογή του Shard κλειδιού επηρεάζει την απόδοση, την αποδοτικότητα και την επεκτασιμότητα ενός sharded cluster. Ένα cluster με το καλύτερο δυνατό υλικό (hardware) και υποδομή (infrastructure) μπορεί να παρεμποδιστεί η απόδοσή του από την επιλογή του Shard κλειδιού. Η επιλογή του Shard κλειδιού και του αντίστοιχου index μπορεί επίσης να επηρεάσει τη sharding στρατηγική που μπορεί να χρησιμοποιήσει το cluster μας.

2.2.4.4 Chunks

Η MongoDB διχοτομεί τα sharded δεδομένα σε κομμάτια (chunks). Κάθε chunk έχει ένα χαμηλό εύρος από *1 έως n* και ένα ανώτερο εύρος *1 έως n-1*, βασισμένο στο shard κλειδί.

Η MongoDB μεταφέρει τα chunks μεταξύ των shards στο sharded cluster με τη χρήση του μηχανισμού εξισορρόπησης συμπλέγματος (sharded cluster balancer). Ο εξισορροπητής (balancer) προσπαθεί να επιτύχει την ίση διανομή των chunks σε όλα τα shards του cluster.

Μέγεθος Chunk

Το προεπιλεγμένο μέγεθος ενός chunk στη MongoDB είναι 64 MB (MegaByte). Μπορείτε να αυξήσετε ή να μειώσετε το μέγεθος του chunk το οποίο όμως θα έχει κάποιες επιπτώσεις που παρατίθενται παρακάτω:

1. Chunks μικρότερα από 64 MB, οδηγούν σε μια πιο ομοιόμορφη κατανομή δεδομένων σε βάρος συχνότερων μετεγκαταστάσεων/μεταφοράς chunks μεταξύ των shards (migrations). Αυτό δημιουργεί περισσότερο φόρτο στο δρομολογητή mongos.

2. Chunks μεγαλύτερα από 64 MB, οδηγούν σε λιγότερες μετεγκαταστάσεις. Αυτό είναι πιο αποτελεσματικό τόσο από την μεριά δικτύωσης όσο και στην αποφυγή εμφάνισης υψηλού επίβαρου (overhead) στο επίπεδο δρομολόγησης ερωτημάτων (mongos). Αλλά, αυτή η αποτελεσματικότητα έρχεται σε βάρος μιας δυνητικά άνισης κατανομής δεδομένων.
3. Το μέγεθος chunk επηρεάζει τον μέγιστο αριθμό εγγράφων (documents) ανά chunk που είναι για να μετεγκατασταθούν.
4. Το μέγεθος του chunk επηρεάζει το μέγιστο μέγεθος συλλογής (collection size) όταν εφαρμόζουμε sharding σε μια υπάρχουσα συλλογή (collection). Μόλις πραγματοποιηθεί το sharding, το μέγεθος του chunk δεν περιορίζει το μέγεθος της συλλογής (collection size).

Περιορισμοί

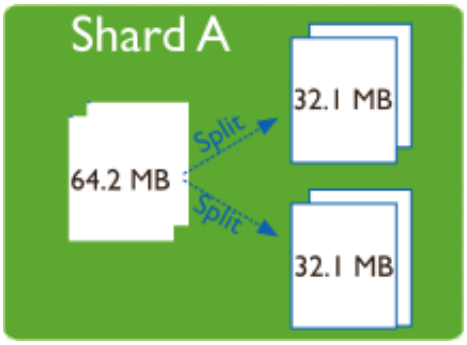
Η αλλαγή του μεγέθους chunk επηρεάζει τη διαδικασία διάσπασης chunks, προσθέτοντας κάποιους περιορισμούς.

- Η αυτόματη διάσπαση chunks γίνεται μόνο κατά την εισαγωγή (insert) ή την τροποποίηση (update) στη βάση. Εάν μειώσετε το μέγεθος του chunk, μπορεί να χρειαστεί χρόνος για να διασπαστούν όλα τα chunks στο νέο μέγεθος.
- Η διάσπαση των chunks δεν μπορεί να "ακυρωθεί". Εάν αυξήσετε το μέγεθος του chunk, τα υπάρχοντα chunks πρέπει να "μεγαλώσουν" μέσα από τις εντολές εισαγωγής (insert) ή τροποποίησης (update) μέχρι να φτάσουν στο νέο προκαθορισμένο μέγεθος.

Διάσπαση Chunks

Η διάσπαση είναι μια διαδικασία που κρατά τα chunks από το να γίνουν αρκετά μεγάλα. Όταν ένα chunk ξεπεράσει ένα προκαθορισμένο μέγεθος chunk ή αν ο αριθμός των εγγράφων (documents) στο chunk υπερβεί το μέγιστο αριθμό εγγράφων ανά τεμάχιο για τη μετεγκατάσταση (Maximum Number of Documents Per Chunk to Migrate), η MongoDB χωρίζει το chunk με βάση τις τιμές του shard κλειδιού που αντιπροσωπεύει το συγκεκριμένο chunk. Ένα chunk μπορεί να χωριστεί σε πολλαπλά chunks όπου χρειάζεται. Οι εντολές εισαγωγής (insert) ή

τροποποίησης (update) μπορούν να προκαλέσουν διάσπαση chunks. Όταν πραγματοποιούνται διασπάσεις chunks, η MongoDB δεν μετεγκαθιστά δεδομένα και δεν επηρεάζει τα shards.



Σχήμα 2.2 Διάσπαση Chunks

Οι διασπάσεις chunks μπορούν να οδηγήσουν σε ανομοιόμορφη κατανομή των chunks για μια συλλογή (collection) μεταξύ των shards. Σε τέτοιες περιπτώσεις, ένα στιγμιότυπο (instance) του mongos ξεκινάει ένα γύρο μετεγκαταστάσεων, για να αναδιανείμει τα chunks ανάμεσα στα shards.

2.2.4.5 Πλεονεκτήματα του Sharding

Αναγνώσεις / Εγγραφές (Reads / Writes)

Το MongoDB διανέμει το φόρτο εργασίας ανάγνωσης (read) και εγγραφής (write) στα shards του sharded cluster, επιτρέποντας σε κάθε shard να επεξεργάζεται ένα υποσύνολο λειτουργιών του cluster. Ο φόρτος εργασίας ανάγνωσης και εγγραφής μπορεί να επεκταθεί οριζόντια (scale out) σε όλο το cluster με την προσθήκη περισσότερων shards.

Για ερωτήματα που περιλαμβάνουν το shard κλειδί ή το πρόθεμα ενός σύνθετου (compound) shard κλειδιού, το mongos μπορεί να δρομολογήσει το ερώτημα σε ένα συγκεκριμένο shard ή σε ένα σύνολο shards. Με αυτό το τρόπο δεν ερωτούνται όλα τα shards του συστήματος, αλλά μόνο αυτά που χρειαζόμαστε, το οποίο αυξάνει την αποδοτικότητα του συστήματος.

Χωρητικότητα αποθήκευσης (Storage Capacity)

Το Sharding διανέμει δεδομένα μεταξύ των shards του cluster, επιτρέποντας σε κάθε shards να περιέχει ένα υποσύνολο των συνολικών δεδομένων του cluster. Καθώς αυξάνεται το σύνολο δεδομένων, επιπλέον shards αυξάνουν την χωρητικότητα αποθήκευσης του cluster.

Μεγάλη διαθεσιμότητα (High Availability)

Ένα sharded cluster μπορεί να συνεχίσει να εκτελεί εν μέρη λειτουργίες ανάγνωσης / εγγραφής ακόμη και αν δεν είναι διαθέσιμο ένα ή περισσότερα shards. Ενώ το υποσύνολο δεδομένων που βρίσκεται στα μη διαθέσιμα shards, δεν είναι προσβάσιμο κατά το χρόνο μη λειτουργίας (downtime), οι αναγνώσεις ή εγγραφές που κατευθύνονται στα διαθέσιμα shards μπορούν να πετύχουν.

2.2.4.6 Εκτίμηση Sharding

Οι απαιτήσεις και η πολυπλοκότητα της υποδομής ενός sharded cluster, απαιτεί προσεκτικό προγραμματισμό, εκτέλεση και συντήρηση.

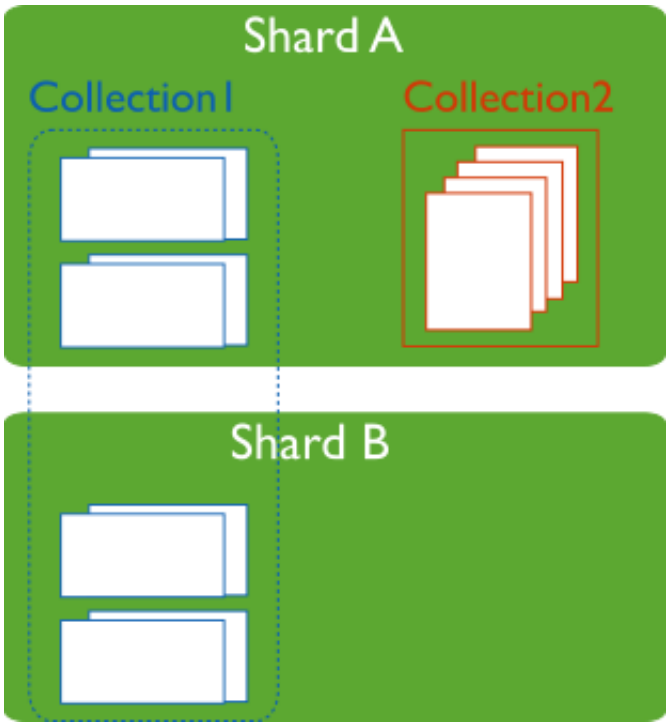
Η προσεκτική ανάλυση στην επιλογή του σωστού shard κλειδιού είναι απαραίτητη για την εξασφάλιση της απόδοσης και της αποδοτικότητας του cluster. Δεν είναι εφικτή η αλλαγή του shard κλειδιού μόλις πραγματοποιηθεί το sharding, ούτε γίνεται να αντιστρέψετε το sharding που έχει πραγματοποιηθεί σε μια sharded συλλογή (collection).

Εάν τα ερωτήματα (queries) δεν περιλαμβάνουν το shard κλειδί ή το πρόθεμα ενός σύνθετου shard κλειδιού, το mongos στέλνει τα ερωτήματα σε όλα τα shards στο sharded cluster (broadcasting). Τέτοιου είδους ερωτήματα θα χρειαστούν μεγάλη διάρκεια για τη διεκπεραίωση τους.

Το sharding μπορεί να έχει κάποιες επιπλέον απαιτήσεις και περιορισμούς καθώς βγαίνουν νέες εκδόσεις της MongoDB, και θα πρέπει να μελετηθούν εξετάζοντας το κατάλληλο εγχειρίδιο χρήσης της εκάστοτε έκδοσης.

2.2.4.7 Sharded και μη-Sharded Συλλογές

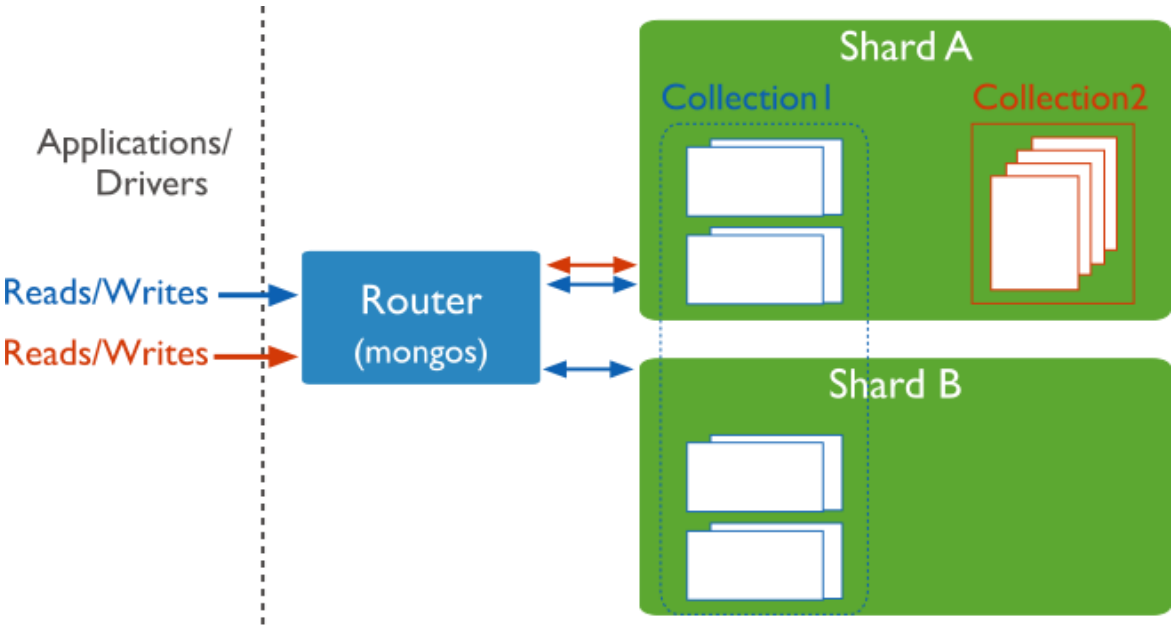
Μια βάση δεδομένων μπορεί να έχει ένα μίγμα από sharded και μη-Sharded συλλογές (non-Sharded Collections). Οι Sharded συλλογές χωρίζονται και διανέμονται στα shards του cluster. Οι μη-Sharded συλλογές αποθηκεύονται σε ένα κύριο shard. Κάθε βάση δεδομένων έχει το δικό της κύριο shard.



Σχήμα 2.3 Sharded και μη-Sharded Συλλογές

2.2.4.8 Σύνδεση με Sharded Cluster

Για να αλληλεπιδράσετε με οποιαδήποτε συλλογή (collection) στο sharded cluster, πρέπει να συνδεθείτε μέσω του δρομολογητή mongos. Αυτό περιλαμβάνει τις sharded και μη-sharded συλλογές. Οι πελάτες (clients) δεν θα πρέπει ποτέ να συνδέονται σε ένα μεμονωμένο shard προκειμένου να εκτελούν λειτουργίες ανάγνωσης ή εγγραφής.



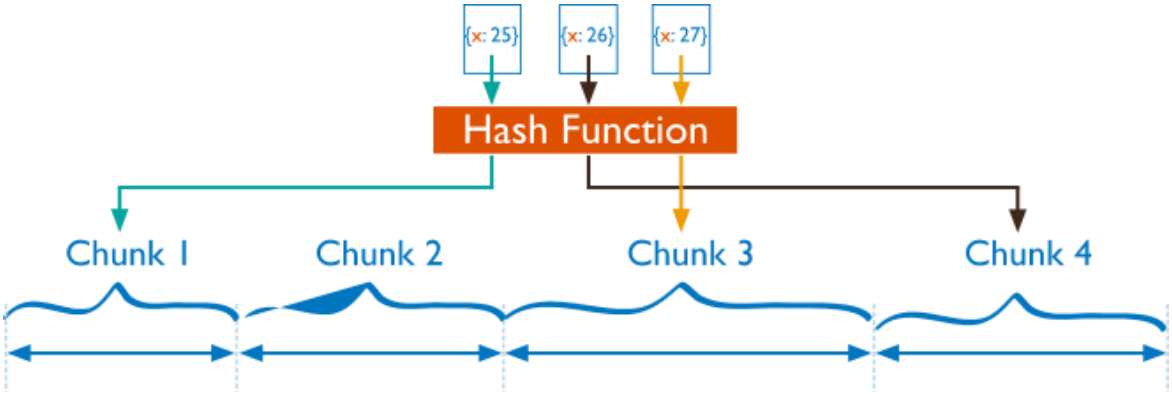
Σχήμα 2.4 Διάγραμμα Σύνδεσης με ένα Sharded Cluster

2.2.4.9 Στρατηγικές Sharding

Η MongoDB υποστηρίζει δύο στρατηγικές sharding για τη διανομή δεδομένων μεταξύ των sharded clusters.

Hashed Sharding

Το Hashed Sharding περιλαμβάνει τον υπολογισμό hash του πεδίου τιμής του Shard κλειδιού. Κάθε Chunk ανατίθεται ένα εύρος βασισμένο στις hashed τιμές του shard κλειδιού.



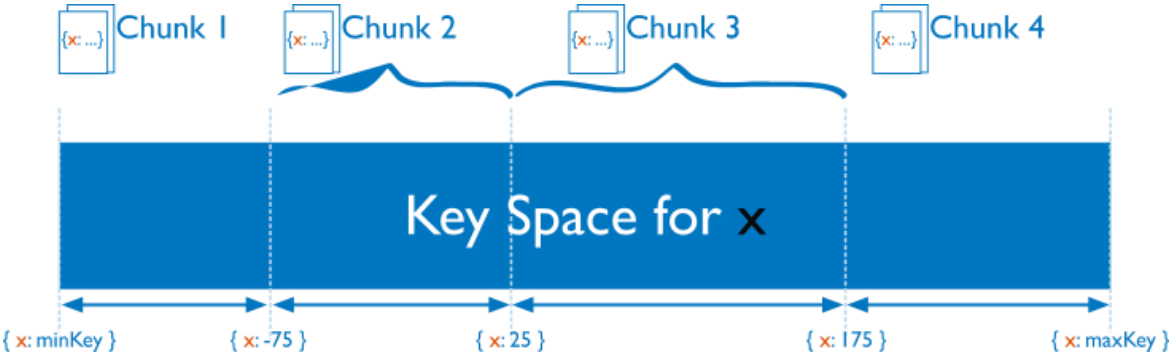
Σχήμα 2.5 Hashed Sharding

Παρόλο που το εύρος από shard κλειδιά μπορεί να είναι "κοντά", οι hash τιμές αυτών είναι απίθανο να βρίσκονται στο ίδιο chunk. Η κατανομή δεδομένων βασισμένη σε hashed τιμές διευκολύνει στο να γίνει η ομοιόμορφη κατανομή δεδομένων.

Ωστόσο, η hashed διανομή σημαίνει ότι ερωτήσεις που βασίζονται σε ένα εύρος τιμών (ranged-based queries) του shard κλειδιού, είναι λιγότερο πιθανό να στοχεύσουν σε ένα ενιαίο shard, με αποτέλεσμα να γίνονται περισσότερα ερωτήματα broadcast μέσα στο cluster.

Ranged Sharding

Το Ranged Sharding περιλαμβάνει τη διαίρεση των δεδομένων σε φάσματα βασισμένο στο shard κλειδί. Έπειτα σε κάθε chunk ανατίθεται ένα εύρος βασισμένο στις τιμές του shard κλειδιού.



Σχήμα 2.6 Ranged Sharding

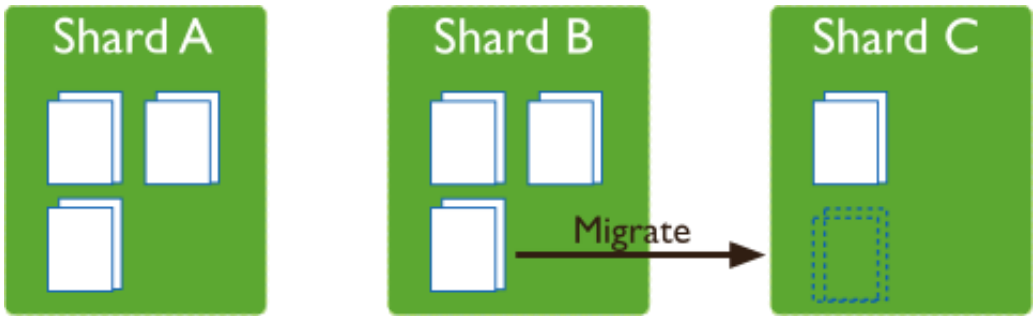
Εάν ένα εύρος από shard κλειδιά των οποίων οι τιμές είναι "κοντά", είναι πιο πιθανό να συνυπάρχουν στο ίδιο chunk. Αυτό επιτρέπει στο mongos να δρομολογεί τα ερωτήματα, μόνο στα shards που περιέχουν τα απαιτούμενα δεδομένα.

Η αποτελεσματικότητα του ranged sharding εξαρτάται από το shard κλειδί που επιλέξατε. Η επιλογή "λάθος" shard κλειδιών μπορεί να οδηγήσει σε άνιση κατανομή δεδομένων, γεγονός που μπορεί να αναιρέσει κάποια από τα οφέλη του sharding ή να προκαλέσει σημεία συμφόρησης (bottlenecks).

2.2.4.10 Εξισορροπητής Sharded Cluster

Ο εξισορροπητής (balancer) της MongoDB, είναι μια διαδικασία υποβάθρου (background) που παρακολουθεί τον αριθμό των chunks σε κάθε shard. Όταν ο αριθμός των chunks σε ένα shard φτάσει σε ένα συγκεκριμένο όριο μεταγκατάστασης (migration threshold), ο εξισορροπητής προσπαθεί να μετακινήσει αυτόματα τα chunks μεταξύ των shards και να φτάσει σε ίσο αριθμό chunks ανά shard.

Η διαδικασία εξισορρόπησης των sharded clusters είναι απολύτως "κρυφή" από το χρήστη και την εφαρμογής, αν και μπορεί να υπάρξει κάποια επίπτωση στην απόδοση του συστήματος κατά τη διάρκεια της εξισορρόπησης.



Σχήμα 2.7 Εξισορροπητής Sharded Cluster

Για να αντιμετωπιστεί η ανομοιογενής κατανομή chunks για μια sharded υλλογή (collection), ο εξισορροπητής μεταφέρει από τα shards με τα περισσότερα chunks,

στα shards που έχουν τα λιγότερα chunks. Ο εξισορροπητής μετακινεί τα chunks, ένα κάθε φορά, μέχρι να υπάρξει μια ομοιόμορφη κατανομή των chunks για τη συλλογή κατά μήκος των shards. [6]

2.3 Τεχνολογία των Containers

2.3.1 Εισαγωγή στους Containers

Για πολλά χρόνια είχαμε μονολιθικές εφαρμογές οι οποίες δούλευαν πολύ καλά σε εικονικές μηχανές (Virtual Machines). Η εισαγωγή των κατανεμημένων συστημάτων, των μικρουπηρεσιών (microservices) και της αυτόματης αυξομείωσης των πόρων των υπολογιστικών συστημάτων δημιούργησε την ανάγκη για αλλαγή στο τρόπο εγκατάστασης και εκτέλεσης του λογισμικού. Αυτό το κενό ήρθε να καλύψει η τεχνολογία των containers. Οι containers είναι πολύ ελαφριά VMs τα οποία εκτελούνται σε ένα Linux περιβάλλον ως μια διεργασία (process). Υπάρχουν αρκετοί διαφορετικοί τύποι από containers (LXC containers, Docker, Rocket) αλλά την μάχη την έχει κερδίσει με διαφορά το Docker, οπότε από εδώ και πέρα όταν θα αναφερόμαστε στη τεχνολογία των containers θα αναφέρεται το Docker.

Το Docker ήρθε και έλυσε δύο μεγάλα προβλήματα στην καθημερινότητα των μηχανικών λογισμικού και όχι μόνο. Το πρώτο πρόβλημα που λύθηκε από τις πρώτες κιόλας εκδόσεις του, είναι ότι βοήθησε τους προγραμματιστές στο να συνδέουν εύκολα και γρήγορα την εφαρμογή τους με βάσεις δεδομένων και με άλλες τεχνολογικές υπηρεσίες που χρειάζεται μία εφαρμογή (συστήματα ουρών, βάσεις δεδομένων, μη σχεσιακές βάσεις δεδομένων). Το δεύτερο, και πολύ πιο σημαντικό, πρόβλημα στο οποίο έδωσε λύση είναι το ότι μπορεί να υποστηρίζει τις δομές των κατανεμημένων συστημάτων, τις μικρουπηρεσίες και τις αυξομειώσεις των πόρων που χρησιμοποιούν οι εφαρμογές, πολύ αποτελεσματικά.

2.3.2 Βασικές έννοιες των Containers

2.3.2.1 Docker Image

Στο Docker όλα ξεκινούν από το docker image. Είναι το πακέτο το οποίο περιέχει το λειτουργικό, την εφαρμογή και τις βιβλιοθήκες που χρειάζεται η εφαρμογή. Σε επίπεδο υλοποίησης είναι ένα σύνολο αρχείων και καταλόγων τα οποία το docker τα παρουσιάζει ως ένα ομοιόμορφο και ολοκληρωμένο σύνολο. Το μικρότερο κομμάτι ενός docker image είναι το επίπεδο (layer). Ένα docker image αποτελείται από ένα ή παραπάνω επίπεδα. Αυτά τα επίπεδα μπορούν να χρησιμοποιηθούν και από άλλα images. Το κάθε docker image περιγράφεται από ένα αρχείο το Dockerfile. Κάθε γραμμή του Dockerfile είναι ένα επίπεδο (layer).

Περιεχόμενα του dockerfile:

```
FROM ubuntu
RUN apt-get update
RUN apt-get install -y nginx
ENTRYPOINT ["/usr/sbin/nginx","-g","daemon off;"]
```

1. **FROM ubuntu** : Κατέβασμα του docker image: Ubuntu από το το κεντρικό αρχείο Docker Hub
2. **RUN apt-get update** : Εκτέλεση της εντολής
3. **RUN apt-get install -y nginx** : Εγκατάσταση του proxy nginx
4. **ENTRYPOINT ["/usr/sbin/nginx ", "-g","daemon off; "]** : Η εντολή που θα τρέξει κατά την εκτέλεση ενός container

Τρέχοντας την παρακάτω εντολή στον φάκελο στον οποίο βρίσκεται το Dockerfile παράγεται το docker image με όνομα *my-nginx*.

```
docker build –t my-nginx
```

2.3.2.2 Docker Container

Το εκτελέσιμο στιγμιότυπο ενός docker image είναι το docker container. Ένα container έχει το δικό του δίκτυο, σύστημα αρχείων, μεταβλητές περιβάλλοντος και είναι πλήρως απομονωμένο από οποιαδήποτε άλλη διεργασία εκτελείται στο ίδιο υπολογιστικό σύστημα. Κάθε docker container έχει το δικό του λειτουργικό σύστημα, μπορεί να έχει το δικό του δίκτυο και είναι απομονωμένο σχεδόν από οτιδήποτε άλλο εκτελείται στο σύστημα. Τα μόνα που μοιράζεται με το λειτουργικό στο οποίο εκτελείται είναι ο πυρήνας (kernel) και τα cgroups. [15]

ΚΕΦΑΛΑΙΟ 3

3. ΥΛΟΠΟΙΗΣΗ

Στην παρούσα ενότητα περιγράφονται οι τρόποι με τους οποίους υλοποιήσαμε την εφαρμογή μας, αναλύοντας τα διάφορα τμήματα του πηγαίου κώδικα της εφαρμογής και τα διάφορα scripts που χρησιμοποιήθηκαν.

3.1 MongoDB Συλλογές

Για τις ανάγκες της εφαρμογής μας, έχουμε δημιουργήσει τις τρεις παρακάτω συλλογές (collections):

3.1.1 Συλλογή posts

Αυτή η συλλογή (collection) περιλαμβάνει όλα τα άρθρα του blog μας. Παρακάτω βλέπουμε στο Σχήμα 8, ένα στιγμιότυπο του blog μας, που περιλαμβάνει ένα άρθρο από το χρήστη John Doe, και ένα σχόλιο από τον χρήστη Bob Johnson.

Σχήμα 3.1 Στιγμιότυπο ενός άρθρου στο blog μας

Το JSON document που αντιστοιχεί στο άρθρο του παραπάνω σχήματος (Σχήμα 3.1) έχει τη παρακάτω μορφή:

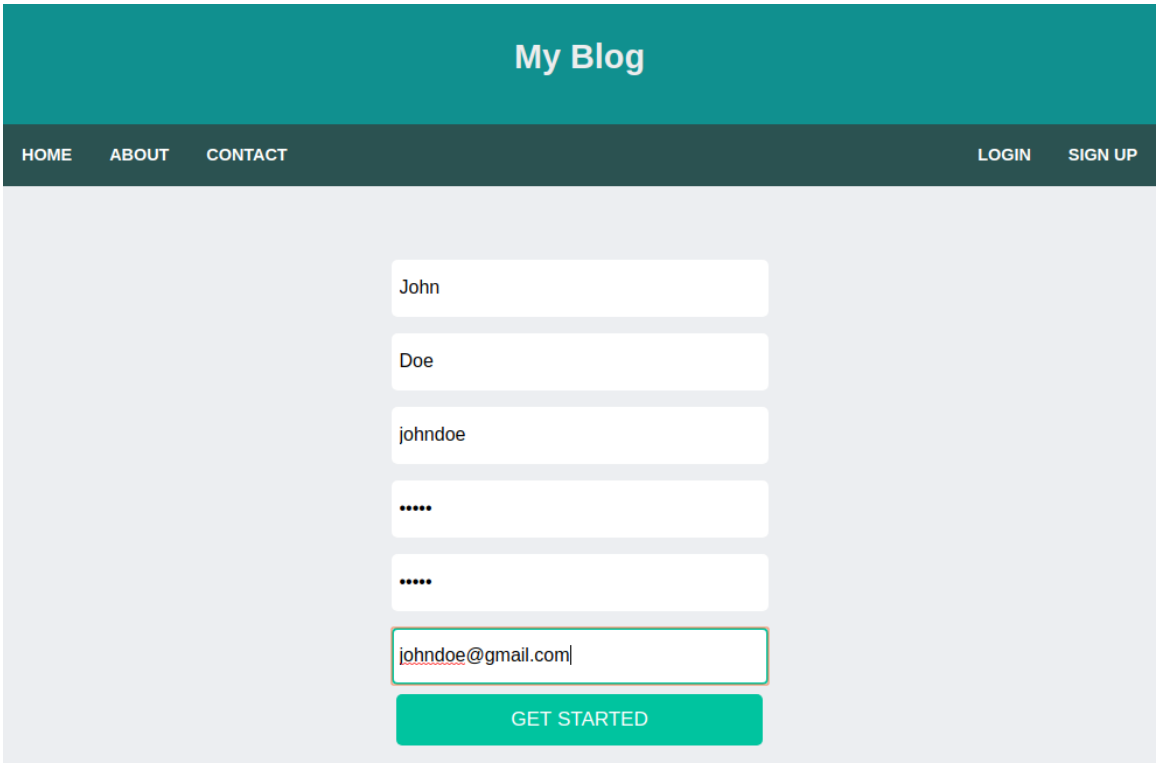
```
db.posts.find({ title : "This My New Post" }).pretty()
{
        "_id" : ObjectId("5af94a4d1f446515e2c2c095"),
        "title" : "This My New Post",
        "author" : "johndoe",
        "body" : "This is the body of my post.",
        "permalink" : "this_my_new_post",
        "tags" : [
                "New",
                "Post"
        ],
        "comments" : [
                {
                        "author" : "Bob Johnson",
                        "body" : "This is a comment by Bob.",
```

```
                        "email" : "bob@gmail.com",
                        "num_likes" : 1
                    }
        ],
        "date" : ISODate("2018-05-14T08:35:25.179Z")
}
```

- **_id**: μοναδικό αναγνωριστικό πεδίο (unique identifier field), για να μπορούμε να ξεχωρίσουμε τα έγγραφα (δημιουργείται αυτόματα από τη MongoDB όταν δεν δώσουμε κάποια τιμή).
- **title**: ο τίτλος του άρθρου.
- **author**: ο συγγραφές του άρθρου.
- **body**: περιλαμβάνει όλο το κύριο κείμενο του άρθρου.
- **permalink**: Το Permalink είναι ένας μόνιμος σύνδεσμος "URL", προς το άρθρο.
- **tags**: Είναι ένα πίνακας (array) που περιλαμβάνει ετικέτες που έχει αναθέσει ο συγγραφέας του άρθρου, στη δική μας περίπτωση, “New”, “Post”.
- **comments**: Είναι ένας πίνακας (array) που περιλαμβάνει όλα τα σχόλια που έχουν γίνει στο άρθρο, αποθηκεύοντας παράλληλα τα στοιχεία του χρήστη που έκανε το σχόλιο.
    - **author**: ο χρήστης που έκανε το σχόλιο, περιλαμβάνει το ονοματεπώνυμό του.
    - **body**: περιλαμβάνει το περιεχόμενο του σχόλιου.
    - **email**: περιλαμβάνει την ηλεκτρονική διεύθυνση ταχυδρομείου του χρήστη.
    - **num_likes**: είναι ο αριθμός των χρηστών στων οποίων άρεσε το συγκεκριμένο σχόλιο.
- **date**: είναι η ημερομηνία και ώρα κατά την οποία δημιουργήθηκε το συγκεκριμένο άρθρο.

3.1.2 Συλλογή users

Αυτή η συλλογή (collection) περιλαμβάνει όλους τους χρήστες που έχουν κάνει εγγραφή στην εφαρμογή μας. Παρακάτω βλέπουμε στο Σχήμα 9 ένα στιγμιότυπο της φόρμας εγγραφής νέου χρήστη.



Σχήμα 3.2 **Στιγμιότυπο της φόρμας εγγραφής νέου χρήστη**

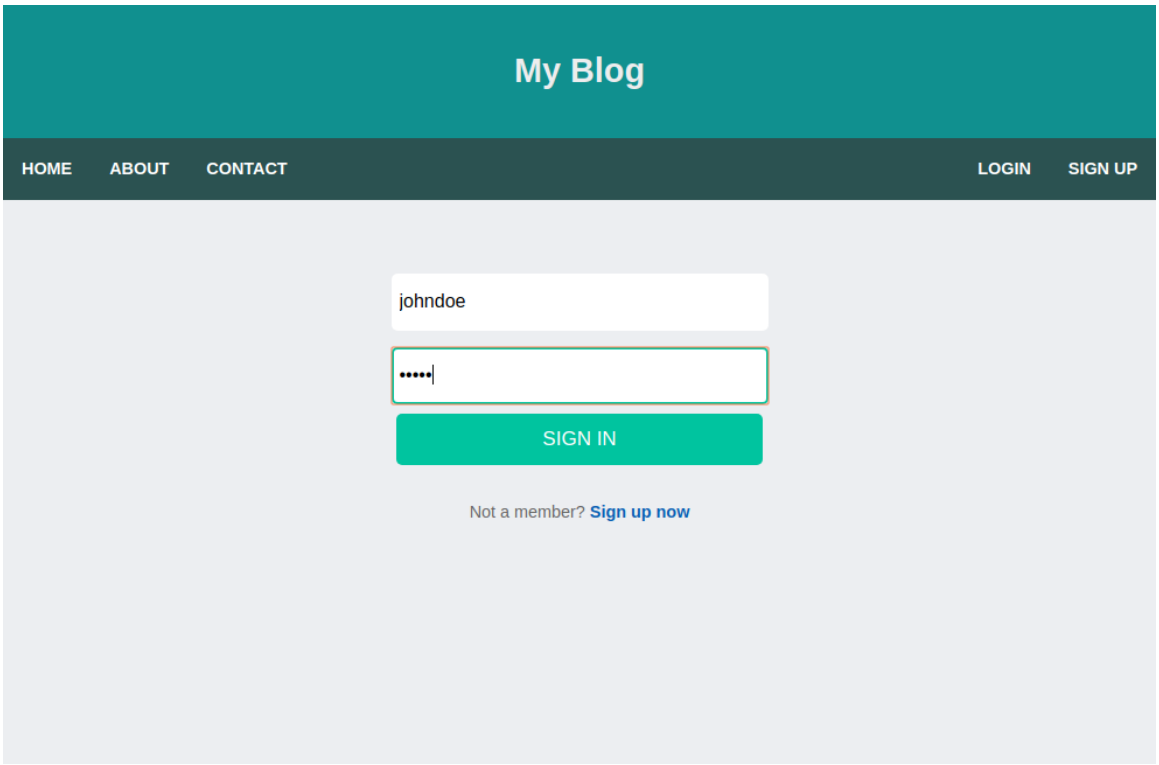
Το JSON document που αντιστοιχεί στο άρθρο του παραπάνω σχήματος (Σχήμα 3.2) έχει τη παρακάτω μορφή:

```
> db.users.find({ _id : "johndoe" }).pretty()
{
    "_id":"johndoe",
    "firstName":"John",
    "lastName":"Doe",
    "password":"77+9JGrvv706FWgsG++/vUDvv70P77+9ems=,1506455398",
    "email":"johndoe@gmail.com"
}
```

- **_id**: περιλαμβάνει το όνομα χρήστη που επέλεξε κατά την εγγραφή του ο χρήστης στην εφαρμογή (πρέπει να είναι μοναδικό).
- **firstName**: περιλαμβάνει το όνομα του χρήστη.
- **lastName**: περιλαμβάνει το επίθετο του χρήστη.
- **password**: περιλαμβάνει την hashed έκδοση του συνθηματικού του χρήστη
- **email**: περιλαμβάνει την ηλεκτρονική διεύθυνση ταχυδρομείου του χρήστη.

3.1.3 Συλλογή sessions

Αυτή η συλλογή (collection) περιλαμβάνει τα sessions των χρηστών. Όταν ένας εγγεγραμμένος χρήστης συνδέεται στην εφαρμογή χρησιμοποιώντας το συνθηματικό του (βλέπε σχήμα 10), δημιουργείται ένα cookie το οποίο και αποθηκεύεται στην συγκεκριμένη συλλογή.



Σχήμα 3.3 Φόρμα σύνδεσης για εγγεγραμένους χρήστη

Εφόσον πραγματοποιηθεί επιτυχώς η σύνδεση του χρήστη στο σύστημα, θα μεταφερθεί (redirect) στη σελίδα υποδοχής (welcome).

Σχήμα 3.4 Στιγμιότυπο της σελίδας υποδοχής

Το JSON document που θα δημιουργηθεί στη sessions συλλογή (collection), έχει τη παρακάτω μορφή:

```
> db.sessions.find({ username : "johndoe" }).pretty()
{
        "_id" : "0wH+yIeKw0NVMIn2LMA5GA3Z8xvl+LIMlTtC8hzrIaw=",
        "username" : "johndoe"
}
```

- **_id**: περιλαμβάνει το cookie του session (είναι μοναδικό)
- **username**: περιλαμβάνει το όνομα χρήστη, του συνδεδεμένου χρήστη

3.2 Η Γλώσσα Προγραμματισμού Java

3.2.1 Περιγραφή της Java

Η Java είναι μία αντικειμενοστραφής γλώσσα προγραμματισμού δομημένη σε κλάσεις που σχεδιάστηκε από την εταιρεία πληροφορικής Sun Microsystems το 1995. Ένα από τα βασικά προτερήματα της Java είναι ό,τι εφαρμογές που είναι γραμμένες σε Java, ο μεταγλωτισμένος κώδικας αυτών (γνωστός ως bytecode), είναι εκτελέσιμος ακριβώς το ίδιο σε Windows, Linux, Unix και Macintosh, χωρίς να χρειαστεί κάποια αλλαγή στο πηγαίο κώδικα για κάθε διαφορετικό λειτουργικό σύστημα.

Μερικά από τα χαρακτηριστικά της Java που τη κάνουν να ξεχωρίζει είναι:

- Είναι απλή, δεν έχει "δύσκολα" χαρακτηριστικά όπως οι δείκτες (pointers) ή οι δομές και ενώσεις (structures και unions) της C.
- Είναι αντικειμενοστραφής (object oriented). Χρησιμοποιεί τις κλάσεις (classes) για να οργανώσει τον κώδικα σε λογικές ενότητες.
- Είναι μεταγλωττιζόμενη (compiled) αλλά μπορεί να χαρακτηριστεί και ως διερμηνευόμενη (interpreted).
- Είναι ασφαλής.
- Υποστηρίζει πολυνημάτωση (multithreading). [9]

3.2.2 Περιγραφή του Spark framework

Το Spark είναι ένα δωρεάν και ελεύθερου λογισμικού (open source) application framework γραμμένο σε Java. Έχει σχεδιαστεί για να είναι εύκολη και γρήγορη η δημιουργία Διεπαφών Προγραμματισμού Εφαρμογών (APIs – Application Programming Interface). Είναι μια "ελαφριά" βιβλιοθήκη που συνδέουμε στην εφαρμογή μας για να αρχίσουμε να σερβίρουμε δεδομένα. Μας επιτρέπει να ορίσουμε διαδρομές (routes) και να τις αποστείλουμε στις λειτουργίες που θα ανταποκριθούν όταν αυτές οι διαδρομές ζητηθούν.

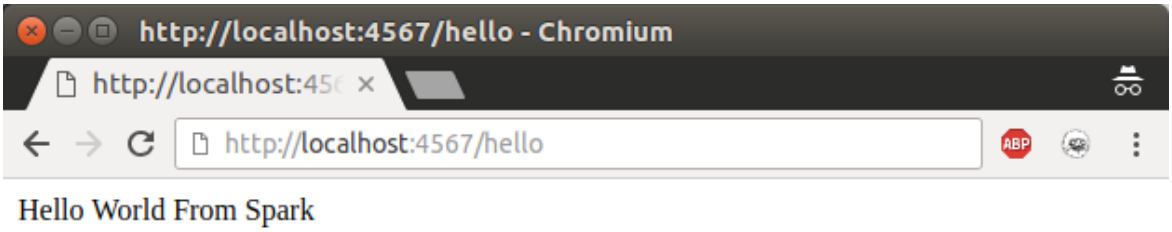
Ένα απλό παράδειγμα χρήσης του Spark framework φαίνεται παρακάτω:

```
import static spark.Spark.*;

public class HelloWorld {
        public static void main(String[] args) {
                Spark.get(new Route("/hello") {
                        @Override
                        public Object handle(final Request request, final Response response
          {
             return "Hello World From Spark";
          }
       });
        }
}
```

Όταν εκτελεστεί αυτή η κύρια (main) μέθοδος, το Spark θα σηκώσει έναν εξυπηρετητή Web (Web server), τον οποίο μπορείτε να επισκεφτείτε ανοίγοντας τον περιηγητής ιστού (browser) σας, και γράφοντας στη μπάρα διευθύνσεων: [10]

http://localhost:4567/hello



Σχήμα 3.5 Στιγμιότυπο Spark framework

3.2.3 Περιγραφή του FreeMarker framework

Το FreeMarker είναι μια μηχανή προτύπου (template engine) βασισμένη σε Java που μπορεί να χρησιμοποιηθεί σε προγράμματα Java.

Στο FreeMarker ορίζετε πρότυπα (templates), τα οποία είναι αρχεία κειμένου (text files) που περιέχουν την επιθυμητή έξοδο (HTML σελίδες, e-mails κλπ), με την εξαίρεση ότι περιέχουν μεταβλητές ${name} και ακόμη υποστηρίζουν δομές επιλογής, επανάληψης όπως επίσης αριθμητικές πράξεις, macros και πολλά ακόμα. Στο πηγαίο κώδικα της εφαρμογής μας, Java, παρέχουμε τις πραγματικές τιμές για τις μεταβλητές που βρίσκονται στα FreeMarker αρχεία και η τελική έξοδος παράγεται βάση αυτής της εισόδου.

Για παράδειγμα, έστω ότι έχουμε το παρακάτω FreeMarker πρότυπο:

```
<html>
......
Hello ${name}!
......
</html>
```

και έστω ότι έχουμε το παρακάτω αντικείμενο στο Java κώδικα μας:

```
......
model.setName("World")
......
```

αυτό θα μας παράγει το παρακάτω HTML αρχείο:

```
<html>
......
Hello World!
......
</html>
```

το οποίο και θα εμφανιστεί στην ιστοσελίδα μας. [10]

3.2.4 Περιγραφή του Maven

Το Maven (Γερμανοεβραϊκή λέξη που σημαίνει συσσωρευτής γνώσης), είναι εργαλείο διαχείρισης έργων λογισμικού (software project management tool). Είναι βασισμένο στη δημιουργία ενός μοντέλου αντικειμένου έργου (POM - Project Object Model), το οποίο διαχειρίζεται το χτίσιμο (build) και της εξαρτήσεις (dependencies) ενός έργου (project). [11]

Ακολουθεί παράδειγμα ενός pom.xml αρχείου:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>eu.thesis</groupId>
        <artifactId>blog</artifactId>
        <version>1.0-SNAPSHOT</version>

        <dependencies>
            <dependency>
                <groupId>org.mongodb</groupId>
                <artifactId>mongo-java-driver</artifactId>
                <version>3.2.2</version>
            </dependency>
        </dependencies>
</project>
```

- **groupId** - θα προσδιορίσει το έργο (project) σας με μοναδικό τρόπο ανάμεσα σε όλα τα έργα (projects) σας.
- **artifactId** - είναι το όνομα του .jar αρχείου της εφαρμογής, χωρίς την έκδοση.
- **version** - αφορά τη τρέχουσα έκδοση της εφαρμογής. [12]

3.2.5 Περιγραφή Κλάσεων/Αρχείων εφαρμογής

Γενικά, ο κώδικας της εφαρμογής αποτελείται από αρχεία Java (τα περισσότερα) για τμήμα του «Ελεγκτή/Μοντέλο» (Controller/Model), FreeMarker Java Template αρχεία για το τμήμα της «Προβολής» (View), προσεγγίζοντας τα πρότυπα αρχιτεκτονικής λογισμικού MVC (Model – View – Controller).

Σε αυτή την ενότητα θα αναλύσουμε με περισσότερη λεπτομέρεια τον ρόλο που αποσκοπεί η κάθε κλάση/αρχείο.

Ο φάκελος /scr/main/java/blog/ περιέχει όλες τις κλάσεις (classes) οι οποίες είναι γραμμένες στην γλώσσα Java. Αναλυτικότερα, οι κλάσεις περιγράφονται ως εξής:

- **BlogController.java** - Περιλαμβάνει τους controllers της εφαρμογής μας. Είναι υπεύθυνη για την διεπαφή με το χρήστη και γενικότερα με τη λειτουργικότητα της εφαρμογής. Επίσης, αναθέτει την οποιαδήποτε επικοινωνία με τη MongoDB σε τρία Αντικείμενα Πρόσβασης Δεδομένων (Data Access Objects – DAOs): BlogPostDAO, SessionDAO και UserDAO.
- **BlogPostDAO.java** - Είναι υπεύθυνη για τη διαχείριση των άρθρων (blog posts), δηλαδή, ανεύρεση άρθρου, δημιουργία άρθρου, προσθήκη σχόλιου και “Μου αρέσει” (like).
- **SessionDAO.java** - Είναι υπεύθυνη για τη διαχείριση των sessions (περίοδος σύνδεσης). Κάθε session περιγράφει ένα σύνολο αλληλεπιδράσεων (interactions) του χρήστη που πραγματοποιούνται στην ιστοσελίδα/εφαρμογή μας, μέσα σε ένα δεδομένο χρονικό διάστημα. Κατά τη διάρκεια ενός session ο επισκέπτης έχει τη δυνατότητα να πραγματοποιήσει ένα σύνολο δραστηριοτήτων όπως προβολή σελίδων, δημιουργία νέου άρθρου, σχόλιου, το μαρκάρισμα ενός άρθρου/σχόλιου ως “Μου αρέσει” (like).
- **UserDAO.java** - Είναι υπεύθυνη για τη διαχείριση των δεδομένων των χρηστών (user data), δηλαδή, ανεύρεση πληροφοριών χρήστη, δημιουγία χρήστη.

Ο φάκελος /scr/main/resources/freemarker/ περιέχει όλα τα FreeMarker Java Template αρχεία που μαζί με το styles.css, αποτελούν το View κομμάτι της εφαρμογής.

Ο φάκελος /scr/main/resources/ περιέχει το αρχείο ρυθμίσεων (configuration file) mongodb.properties, της MongoDB.

3.2.6 Το Ολοκληρωμένο Περιβάλλον Προγραμματισμού IntelliJ IDEA

Το IntelliJ IDEA είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE - Integrated development environment) για τη γλώσσα προγραμματισμού Java, που χρησιμοποιήθηκε για την ανάπτυξη της εφαρμογής μας.

3.3 Η Γλώσσα Προγραμματισμού Python

Η Python είναι μία γλώσσα προγραμματισμού υψηλού επιπέδου διερμηνευτή γλώσσας, η οποία δημιουργήθηκε από τον Ολλανδό Guido van Rossum το 1990. Το κύριο προτέρημα της γλώσσας είναι η αναγνωσιμότητα του κώδικα αλλά και η ευκολία χρήσης της. Αποτελεί μία πολύ καλή γλώσσα προγραμματισμού για να ξεκινήσει να ασχολείται με τον κλάδο. Η γλώσσα έχει πάρα πολλές δυνατότητες, αλλά δεν απαιτείται η πλήρης κατανόηση και γνώση του συντακτικού για να μπορέσει κάποιος μηχανικός να αναπτύξει λογισμικό. Με λίγα λόγια η Python έχει αρκετά μικρή καμπύλη εκμάθησης (learning curve) σε σχέση με τις υπόλοιπες γλώσσες προγραμματισμού υψηλού επιπέδου. [16]

Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε για την εκπόνηση της αξιολόγησης απόκρισης του συστήματος. Το λογισμικό αυτό υποστηρίζει τρείς διαφορετικούς τρόπους εκτέλεσης. Αρχικά την εκτέλεση εντολών μόνο εγγραφής προς το blog, έπειτα εκτέλεση εντολών ανάγνωσης δεδομένων από το σύστημα και τέλος αναλογικό συνδιασμό εγγραφών και ανγνώσεων δεδομένων.

Write-only: Δημιουργούνται μόνο εγγραφές στο σύστημα.

Read-only: Ανάγνωση δεδομένων.

Read-write: Αναλογικός συνδυασμός των παραπάνω.

```
$ python test.py --help
usage: Run blog tests [-h] [--host HOST] [--data-set-size
DATA_SET_SIZE]
                      [--user USER] [--read-only] [--write-only]
                      [--read-write] [--comment]

optional arguments:
  -h, --help              show this help message and exit
  --host HOST, -H HOST  Blog url.
  --data-set-size DATA_SET_SIZE, -D DATA_SET_SIZE
                          Data set size.
  --user USER, -U USER  User prefix.
  --read-only, -R         Read only.
  --write-only, -W        Write only.
  --read-write, -A        Read-Write.
  --comment, -C          Add comments after every new post.
```

3.4 Docker Compose

Το docker-compose είναι ένα εργαλείο στο οποίο ορίζονται και μπορούν να εκτελεστούν εφαρμογές που αποτελούνται από πολλούς containers. Με το docker-compose, χρησιμοποιείται ένα αρχείο μορφής YAML ώστε να διαμορφωθούν τα επιμέρους στοιχεία της εφαρμογής μας. Έπειτα με μία μόνο εντολή μπορούμε να ξεκινάμε όλα τα services που περιέχει το YAML αρχείο. Τα επιμέρους μέρη του service είναι οι docker containers. Εκτελώντας την εντολή docker-compose up -d, ξεκινούν όλοι οι containers. Με λίγα λόγια το docker-compose είναι ένα εργαλείο για να διαχειριζόμαστε ευκολότερα τους docker containers όπου αποτελούν μία εφαρμογή ή μια υπηρεσία. [17]

3.5 Ansible

Εγκατάσταση της εφαρμογής με την βοήθεια του Ansible. Το Ansible είναι ένα εργαλείο ανοιχτού κώδικα (open source), διαχείρισης διαμόρφωσης λογισμικού (configuration management tool). Είναι γραμμένο σε python, γεγονός που δίνει τη

δυνατότητα να αναπτύσσονται με μεγάλη ευκολία νέες βιβλιοθήκες που εξυπηρετούν τις ανάγκες μας. Το 2015 αγοράστηκε από την RedHat. Το εργαλείο είναι βασισμένο στην σύνταξη YAML. Δεν απαιτεί την εγκατάσταση deamon στα υπολογιστικά συστήματα τα οποία είναι στόχος της εγκατάστασης, όπως συμβαίνει σε αντίστοιχα διάσημα εργαλεία. Η σύνδεση με τους hosts/servers γίνεται μέσω του πρωτοκόλλου SSH. Για την εγκατάσταση του Ansible λοιπόν, απαιτείται να είναι εγκατεστημένα στο μηχάνημα που εκτελείται η python και το OpenSSH. [18] Έπειτα το Ansible μπορεί να εγκατασταθεί με μία εντολή, ως μία python βιβλιοθήκη. Ακολουθούν επεξήγησεις των βασικών εννοιών του Ansible.

- hosts: αρχείο που αποτελείται από τους στόχους (IP/hostname) όλων των ansible εντολών.
- playbook: περιέχει τις παραμέτρους μίας εγκατάστασης. Κάθε playbook αντιστοιχίζει μία ομάδα από hosts σε μία λίστα από ρόλους.
- roles: μία ομάδα από tasks όπου το σύνολο τους παρέχει μία ολοκληρωμένη διαδικασία, π.χ. εγκατάσταση εφαρμογής.
- tasks: είναι μία εντολή που καλείται μέσω των ansible modules, όπου κάθε module είναι μία ansible βιβλιοθήκη.

3.6 HAProxy

Ο HAProxy είναι λογισμικό ανοιχτού κώδικα που προσφέρει ένα σύστημα υψηλής διαθεσιμότητας εξισορρόπησης φόρτου (highly available load balancer) και διακομιστή μεσολάβησης (proxy server) για TCP και HTTP εφαρμογές που επεκτείνεται σε πολλαπλούς διακομιστές. Είναι ανεπτυγμένος στη γλώσσα δομημένου προγραμματισμού C και έχει τη φήμη ότι είναι πολύ γρήγορος και αποτελεσματικός. Έχει αναπτυχθεί το 2000 από τον Willy Tarreau, ο οποίος έχει συνεισφέρει στον Linux Kernel και συνεχίζει να το κάνει μέχρι σήμερα. [19]

Χρησιμοποιήθηκε ώστε να διανείμει τον φόρτο εργασίας των πολλαπλών οντωτήτων των Java εφαρμογών. Ο HaProxy όπως και όλες οι εφαρμογές και οι υπηρεσίες εκτελείται μέσα σε docker container.

3.7 Jetty

Ο Jetty είναι ένας Java HTTP Web server και Java Servlet container. Χρησιμοποιείται για τη διασύνδεση μεταξύ υπολιστικών συστημάτων, και συνήθως βρίσκεται ενσωματωμένο σε άλλα, μεγαλύτερα software frameworks, όπως το Spark.

3.8 Μεθοδολογία εγκατάστασης εφαρμογής

Τεχνολογίες που χρησιμοποιήθηκαν για την εγκατάσταση του MongoDB Cluster, του Java Application και των τεστς:

- Docker
- Docker-compose
- Ansible
- Python

Η εφαρμογή αποτελείται από μία Java web εφαρμογή η οποία χρησιμοποιεί ως βάση δεδομένων την MongoDB. Η Java εφαρμογή είναι ανεπτυγμένη σύμφωνα με το build εργαλείο Apache Maven. Όλες οι εφαρμογές και οι τεχνολογίες πακετάρονται σε docker images.

3.8.1 Java εφαρμογή

Η Java εφαρμογή πακετάρεται μέσα σε ένα docker image, του οποίου το Dockerfile ακοπλουθεί:

```
FROM jetty:jre8

COPY apache-maven-3.3.3-bin.tar.gz /opt/apache-maven-3.3.3-bin.tar.gz

RUN cd /opt && tar -xvf apache-maven-3.3.3-bin.tar.gz && ln -s
apache-maven-3.3.3/bin/mvn mvn && chmod 0700 mvn && mkdir mongo-project
```

```
  RUN apt-get -y update && apt-get -y install software-properties-
common
  RUN apt-get -y install python-software-properties

  RUN mkdir /opt/jdk && cd /opt && \
    wget --header "Cookie: oraclelicense=accept-securebackup-cookie"
http://download.oracle.com/otn-pub/java/jdk/8u151-
b12/e758a0de34e24606bca991d704f6dcbf/jdk-8u151-linux-x64.tar.gz && \
    tar -zxf jdk-8u151-linux-x64.tar.gz -C /opt/jdk && \
    update-alternatives --install /usr/bin/java java
/opt/jdk/jdk1.8.0_151/bin/java 100 && \
    update-alternatives --install /usr/bin/javac javac
/opt/jdk/jdk1.8.0_151/bin/javac 100

  WORKDIR /opt/mongo-project

  COPY mongo-project /opt/mongo-project

  ENV JAVA_HOME="/opt/jdk/jdk1.8.0_151"
  EXPOSE 8082

  CMD /opt/mvn compile exec:java -Dexec.mainClass=blog.BlogController
-Dexec.args="mongodb://mongos:27000" && tail -f /opt/mongo-project/pom.xml
```

https://github.com/chaldeos/thesis/blob/develop/ansible-deployer/roles/blog/templates/Dockerfile.j2

Το docker image της εφαρμογής ξεκινάει από το βασικό docker image `jetty:jre8` έχει ήδη εγκατεστημένο το Jetty. Ακόμη περιέχει τα εξής:

- apache-maven-3.3.3
- jdk1.8.0_151
- java κώδικας

Το χτίσιμο του docker image γίνεται μέσω του Ansible playbook thesis.yml, εκτελώντας τον ρόλο blog:

```
ansible-playbook thesis.yml -i hosts -t build -v
```

Ακολουθεί το σημείο του playbook που εκτελεί τον ρόλο του blog με την παράμετρο action=”build”. Το σημείο αυτό ενεργοποιείται με το tag=”build”.

```
- hosts: blog
  sudo: no
  roles:
   - { role: "blog", action: "build" }
  tags:
   - build
```

Επιλέγοντας τη ρύθμιση tag=blog-cluster, πραγματοποιείται η εγκατάσταση της εφαρμογής σε cluster. Πιο συγκεκριμένα εκτελούνται δύο docker containers της εφαργογής που βρίσκονται πίσω από έναν HAProxy. Ο HAProxy είναι υπεύθυνος για την κατανομή του φόρτου.

```
ansible-playbook thesis.yml -i hosts -t blog-cluster -v
```

3.8.2 MongoDB Cluster

Ακολουθεί το docker image της MongoDB που χρησιμοποιήθηκε ως base docker image για όλα τα κομμάτια του MongoDB Cluster.

https://raw.githubusercontent.com/docker-library/mongo/753d566a83a4e9734227f186e554c87b4f08be51/3.2/Dockerfile

Τα οποία είναι:

- Mongos

- Mongo Config Server
- Mongo Shard Server

3.8.2.1 Mongos

Για την εγκατάσταση του Mongos docker-container χρησιμοποιήθηκε το παρακάτω docker-compose αρχείο. Στο οποίο παρατίθενται τα εξής χαρακτηριστικά της εγκατάστασης:

```
mongos:
  image: mongo:3.2
  container_name: mongos
  hostname: mongos
  net: thesis
  expose:
   - "27017"
  ports:
   - "27000:27017"
  volumes:
   - "/home/yiannis/mongo-config/keyfile:/etc/mongos/keyfile"
   - "/home/yiannis/mongo-config/mongos.conf:/etc/mongos/mongos.conf"
  command: bash -c "mongos --config /etc/mongos/mongos.conf"
```

Πίνακας 3.1 Περιγραφή του mongos docker-compose

Όνομα docker image	**mongo:3.2**
Όνομα docker container	mongos
Όνομα docker δικτύου τύπου bridge	thesis
Πόρτα mongoDB docker image	27000
Docker volume κειδιού	/home/thesis/mongo-config/keyfile
Docker volume αρχείου παραμέτρων	/home/thesis/mongo-config/mongos.conf
Εντολή εκκίνησης του container	bash –c “mongos –config /etc/mongos/mongos.conf”

3.8.2.2 Mongo Config Server

Ακολουθεί το docker-compose αρχείο που χρησιμοποιήθηκε για την εγκατάσταση των Mongo Configs Servers σε docker-containers. Στο οποίο παρατίθενται τα εξής χαρακτηριστικά της εγκατάστασης:

```
mongo-config-1:
  image: mongo:3.2
  container_name: mongo-config-1
  hostname: mongo-config-1
  net: thesis
  expose:
   - "27019"
  ports:
   - "29001:27019"
  volumes:
   - "/home/yiannis/mongo-config/keyfile:/etc/mongod/keyfile"
   - "/home/yiannis/mongo-config/mongod.conf:/etc/mongod/mongod.conf"
```

```
        #- "/var/run/docker.sock:/var/run/docker.sock"
    command: bash -c "mongod --config /etc/mongod/mongod.conf"
```

Πίνακας 3.2 Περιγραφή του mongo config server docker-compose

Όνομα docker image	**mongo:3.2**
Όνομα docker container	mongo-config-{1..3}
Όνομα docker δικτύου τύπου bridge	thesis
Πόρτα mongoDB docker image	29001
Docker volume κειδιού	/home/thesis/mongo-config/keyfile
Docker volume αρχείου παραμέτρων	/home/thesis/mongo-config/mongod.conf
Εντολή εκκίνησης του container	bash -c "mongod --config /etc/mongod/mongod.conf"

3.8.2.3 Mongo Shard Server

Τέλος το docker-compose αρχείο που χρησιμοποιήθηκε για την εγκατάσταση των Mongo Shard Servers σε docker-containers. Το οποίο είναι το εξής:

```
mongo-shard-1:
    image: mongo:3.2
    container_name: mongo-shard-1
    hostname: mongo-shard-1
    net: thesis
    expose:
     - "27018"
    ports:
     - "28001:27018"
    volumes:
```

```
    - "/home/yiannis/mongo-config/keyfile:/etc/mongod/keyfile"
    - "/home/yiannis/mongo-config/mongod-shard.conf:/etc/mongod/mongod-shard.conf"
  command: bash -c "mongod --config /etc/mongod/mongod-shard.conf --replSet rs1"
```

Πίνακας 3.3 Περιγραφή του mongo shard server docker-compose

Όνομα docker image	**mongo:3.2**
Όνομα docker container	mongo-shard-{1..3}
Όνομα docker δικτύου τύπου bridge	thesis
Πόρτα mongoDB docker image	28001
Docker volume κειδιού	/home/thesis/mongo-config/keyfile
Docker volume αρχείου παραμέτρων	/home/thesis/mongo-config/mongod-shard.conf
Εντολή εκκίνησης του container	bash -c "mongod --config /etc/mongod/mongod-shard.conf" --replSet rs1"

Ακολουθούν οι εντολές για την αρχικοποίηση του MongoDB Cluster.

```
docker exec -ti mongo-shard-1 mongo --port 27018 --eval "rs.initiate()"
docker exec -ti mongo-shard-1 mongo --port 27018 --eval "rs.slaveOk()"
docker exec -ti mongo-config-1 mongo --port 27019 --eval "rs.slaveOk()"
docker exec -ti mongos mongo --eval "sh.addShard(\"rs1/mongo-shard-1:27018\")"
docker exec -t mongo-shard-1 mongo --port 27018  --eval "rs.status()"
```

Ακολουθούν οι εντολές για την αρχικοποίηση του MongoDB Cluster με τρείς Sharding Servers. Ο docker container ‘mongo-shard-1’ είναι ο Primary Shard Server, ενώ οι containers mongo-shard-2 και mongo-shard-3, αποτελούν τα

Secondary. Ακόμη αξίζει να σημειωθεί πως ο Primary Shard Server δηλώνεται στον server του Mongos, ενώ οι Secondary servers δηλώνονται στον Primary.

```
docker exec -ti mongo-shard-1 mongo --port 27018 --eval "rs.initiate()"
docker exec -ti mongo-shard-1 mongo --port 27018 --eval "rs.slaveOk()"
docker exec -ti mongo-shard-2 mongo --port 27018 --eval "rs.slaveOk()"
docker exec -ti mongo-shard-3 mongo --port 27018 --eval "rs.slaveOk()"
docker exec -ti mongo-config-1 mongo --port 27019 --eval "rs.slaveOk()"
docker exec -ti mongos mongo --eval "sh.addShard(\"rs1/mongo-shard-1:27018\")"
docker exec -ti mongo-shard-1 mongo --port 27018 --eval "ρσ.add(\"mongo-shard-2:27019\")"
docker exec -ti mongo-shard-1 mongo --port 27018 --eval "ρσ.add (\"mongo-shard-3:27020\")"
```

Έπειτα από την εγκατάσταση του MongoDB Cluster, η κατάσταση του συστήματος περιγράφεται από το αποτέλεσμα της εντολής ‘rs.status()’ σε οποιονδήποτε από τους Shard Servers.

```
MongoDB shell version: 3.2.20
connecting to: 127.0.0.1:27018/test
{
        "set" : "rs1",
        "date" : ISODate("2018-06-04T21:57:33.546Z"),
        "myState" : 2,
        "term" : NumberLong(1),
        "syncingTo" : "mongo-shard-2:27018",
        "heartbeatIntervalMillis" : NumberLong(2000),
        "members" : [
                {
                        "_id" : 0,
                        "name" : "mongo-shard-1:27018",
                        "health" : 1,
                        "state" : 1,
                        "stateStr" : "PRIMARY",
                        "uptime" : 7997,
                        "optime" : {
                                "ts" : Timestamp(1528149451, 11),
                                "t" : NumberLong(1)
                        },
                        "optimeDate" : ISODate("2018-06-04T21:57:31Z"),
                        "lastHeartbeat" : ISODate("2018-06-04T21:57:31.633Z"),
                        "lastHeartbeatRecv" : ISODate("2018-06-04T21:57:31.768Z"),
                        "pingMs" : NumberLong(0),
```

```
                        "electionTime" : Timestamp(1528141454, 2),
                        "electionDate" : ISODate("2018-06-04T19:44:14Z"),
                        "configVersion" : 3
                },
                {
                        "_id" : 1,
                        "name" : "mongo-shard-2:27018",
                        "health" : 1,
                        "state" : 2,
                        "stateStr" : "SECONDARY",
                        "uptime" : 7997,
                        "optime" : {
                                "ts" : Timestamp(1528149451, 11),
                                "t" : NumberLong(1)
                        },
                        "optimeDate" : ISODate("2018-06-04T21:57:31Z"),
                        "lastHeartbeat" : ISODate("2018-06-04T21:57:31.626Z"),
                        "lastHeartbeatRecv" : ISODate("2018-06-
04T21:57:33.534Z"),
                        "pingMs" : NumberLong(0),
                        "syncingTo" : "mongo-shard-1:27018",
                        "configVersion" : 3
                },
                {
                        "_id" : 2,
                        "name" : "mongo-shard-3:27018",
                        "health" : 1,
                        "state" : 2,
                        "stateStr" : "SECONDARY",
                        "uptime" : 8015,
                        "optime" : {
                                "ts" : Timestamp(1528149453, 10),
                                "t" : NumberLong(1)
                        },
                        "optimeDate" : ISODate("2018-06-04T21:57:33Z"),
                        "syncingTo" : "mongo-shard-2:27018",
                        "configVersion" : 3,
                        "self" : true
                }
        ],
        "ok" : 1
}
```


3.8.3 Εγκατάσταση HAProxy

Το Dockerfile που περιγάφει το σύστημα αρχείων του HAProxy είναι το εξής:

```
FROM haproxy:1.8

ADD haproxy.cfg /usr/local/etc/haproxy/haproxy.cfg
```

Αρχείο με τις παραμέτρους του HAProxy, το οποίο κάνει εξισορρόπηση φόρτου (load bancing) τύπου Round Robin, ανάμεσα στους δύο containers του blog, στις πόρτες 8090 και 8091.

```
global
        user root
        group root
        daemon
        log 127.0.0.1 local0 debug

defaults
        log     global
        mode    http
        option  httplog
        option  dontlognull
        retries 3
        option redispatch
        option http-server-close
        option forwardfor
        timeout connect 5s
        timeout client  15min
        timeout server  15min

frontend public
        bind *:80
        acl url_blog path_beg /blog
        use_backend web-backend if url_blog
        default_backend web-backend

backend web-backend
   balance roundrobin
   mode http
   server blog1 localhost:8090 check
```

```
    server blog2 localhost:8091 check
```

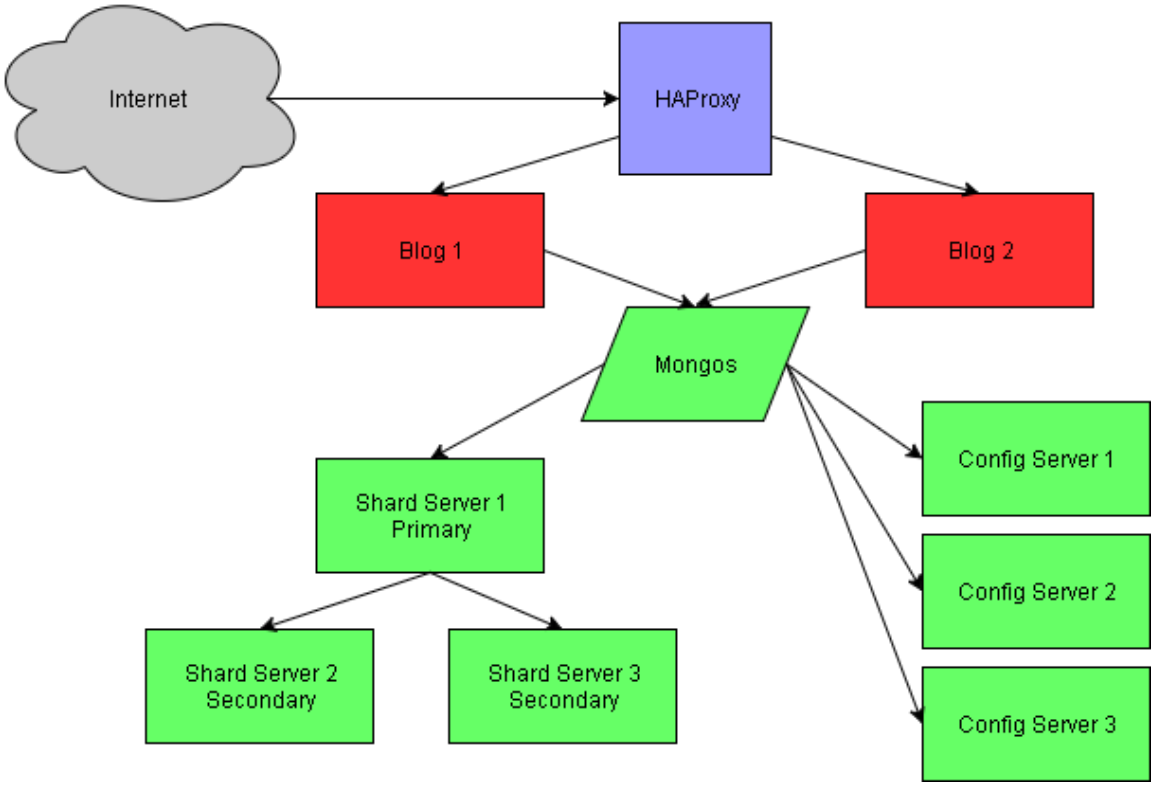
Χτίσιμο του docker image.

```
docker build -t myhaproxy .
```

Η εγκατάσταση του HAProxy πραγματοποιείται με την παρακάτω εντολή:

```
docker run -ti -d -p "8082:80" --name haproxy myhaproxy
```

Τέλος, ακολουθεί το σχήμα της αρχιτεκτονικής του συστήματος το οποίο περίεχει όλες τις παραπάνω τεχνολογίες. Ακόμη συνοψίζει όλες τις επικοινωνίες μεταξύ των κόμβων του συστήματος, του οποίου τα κύρια στοιχεία του είναι το Blog Cluster και το MongoDB Cluster.



Σχήμα 3.6 Αρχιτεκτονική συστήματος

ΚΕΦΑΛΑΙΟ 4

4. ΑΞΙΟΛΟΓΗΣΗ ΑΠΟΔΟΣΗ ΣΥΣΤΗΜΑΤΟΣ

Στο κεφάλαιο αυτό γίνεται η αξιολόγηση του συστήματός μας. Για την αξιολόγηση του συστήματος εκτελέστηκαν τριών διαφορετικών ειδών μεθόδοι εγκατάστασεις και διαχείρισης των δεδομένων σε τρεις διαφορετικές ανάγκες της εφαρμογής. Οι μέθοδοι αφορούν κυρίως την οργάνωση των δεδομένων που αποθηκεύονται στην MongoDB. Αρχικά η αξιολόγηση του συστήματος εκτελέστηκε με τις αρχικές ρυθμίσεις ενός MongoDB Sharded Cluster (χωρίς κάποιο sharded collection). Οι επόμενες δύο μέθοδοι αφορούν τα hashed keys και τα ranged-based keys. Για κάθε μέθοδο εκτελέστηκαν ξεχωριστές αξιολογήσεις συστήματος με εγγραφές δεδομένων, αναγνώσεις δεδομένων καθώς και με αναλογικό συνδιασμό των προαναφερθέντων. Ακόμη, συνδιασμός των μεθόδων και των ενεργείων πάνω στα δεδομένα εκτελέστηκαν με διαφορετικό αριθμό χρηστών (1000, 5000, 10000).

4.1 Sharded Cluster με αρχικές ρυθμίσεις

Σε αυτό το σενάριο, δεν έχουμε καμία sharded συλλογή (collection). Εξ ορισμού, η MongoDB θα τοποθετήσει όλες τις μη-sharded συλλογές (collections), στο Primary shard.

Πίνακας 4.1 Αποτελέσματα Εγγραφής Δεδομένων (χωρίς κάποιο sharded collection)

Εγγραφή Δεδομένων		
Εντολή	**Αριθμός χρηστών**	**Διάρκεια**
python test.py --user dfirstw --data-set-size 400 --prefix a b c d --write-only --parallel 5	10000	0:00:25.782076
python test.py --user dtwow --data-set-size 2000 --prefix a b c d --write-only --parallel 5	50000	0:02:54.944338
python test.py --user dfivew --data-set-size 5000 --prefix a b c d --write-only --parallel 5	125000	0:12:03.552131

Πίνακας 4.2 Αποτελέσματα Ανάγνωσης Δεδομένων (χωρίς κάποιο sharded collection)

Ανάγνωση Δεδομένων		
Εντολή	**Αριθμός χρηστών**	**Διάρκεια**
python test.py --user dfirstr --data-set-size 400 --prefix a b c d --read-only --parallel 5	10000	0:00:21.862046
python test.py --user dtwor --data-set-size 2000 --prefix a b c d --read-only --parallel 5	50000	0:03:50.402784
python test.py --user dfiver --data-set-size 5000 --prefix a b c d --read-only --parallel 5	125000	0:19:00.162130

Πίνακας 4.3 Αποτελέσματα Ανάγνωσης και Εγγραφής Δεδομένων (χωρίς κάποιο sharded collection)

Ανάγνωση και Εγγραφή Δεδομένων		
Εντολή	**Αριθμός χρηστών**	**Διάρκεια**
python test.py --user dfirst --data-set-size 400 --prefix q r s t --read-write --parallel 5	10000	0:00:36.036881
python test.py --user dtwo --data-set-size 2000 --prefix q r s t --read-write --parallel 5	50000	0:04:40.132227
python test.py --user dfive --data-set-size 5000 --prefix q r s t --read-write --parallel 5	125000	0:21:25.374162

4.2 Sharded Cluster με hashed key το _id

Σε αυτό το σενάριο, έχουμε χρησιμοποιήσει τη μέθοδο hashed sharding στη συλλογή (collection) ***posts***. Χρησιμοποιήσαμε ως index το πεδίο *_id*. Παρακάτω παρουσιάζουμε πως γίνεται η δημιουργία του hashed index *_id*, και πως κάνουμε

shard τη posts συλλογή (collection) χρησιμοποιώντας το κλειδί αυτό. Εντός του mongo shell εκτελούμε:

Για να δημιουργήσουμε το index:

```
use blog
db.posts.createIndex( { _id: "hashed" } )
```

Για να ενεργοποιήσουμε τη δυνατότητα να γίνει sharding στη βάση *blog*:

```
use admin
db.runCommand( { enableSharding: "blog" } )
```

Για να κάνουμε shard τη συλλογή posts:

```
db.runCommand( { shardCollection: "blog.posts", key: { _id: "hashed" },
unique: false } )
```

Πίνακας 4.4 Αποτελέσματα Εγγραφής Δεδομένων (με Hashed Key “_id”)

Εγγραφή Δεδομένων		
Εντολή	**Αριθμός χρηστών**	**Διάρκεια**
python test.py --user hfirstw --data-set-size 400 --prefix a b c d --write-only --parallel 5	10000	0:00:25.989883
python test.py --user htwow --data-set-size 2000 --prefix a b c d --write-only --parallel 5	50000	0:02:56.739184
python test.py --user hfivew --data-set-size 5000 --prefix q r s t --write-only --parallel 5	125000	0:12:32.910953

Πίνακας 4.5 Αποτελέσματα Ανάγνωσης Δεδομένων (με Hashed Key "_id")

Ανάγνωση Δεδομένων		
Εντολή	**Αριθμός χρηστών**	**Διάρκεια**
python test.py --user hfirstr --data-set-size 400 --prefix a b c d --read-only --parallel 5	10000	0:00:23.714787
python test.py --user htwor --data-set-size 2000 --prefix a b c d --read-only --parallel 5	500000	0:03:54.943827
python test.py --user hfiver --data-set-size 5000 --prefix a b c d --read-only --parallel 5	125000	0:19:39.380095

Πίνακας 4.6 Αποτελέσματα Ανάγνωσης και Εγγραφής Δεδομένων (με Hashed Key "_id")

Ανάγνωση και Εγγραφή Δεδομένων		
Εντολή	**Αριθμός χρηστών**	**Διάρκεια**
python test.py --user hfirst --data-set-size 400 --prefix q r s t --read-write --parallel 5	10000	0:00:38.09273
python test.py --user htwo --data-set-size 2000 --prefix q r s t --read-write --parallel 5	50000	0:04:50.249525
python test.py --user hfive --data-set-size 5000 --prefix q r s t --read-write --parallel 5	125000	0:21:40.494806

4.3 Sharded Cluster με hashed key το permalink

Σε αυτό το σενάριο, έχουμε χρησιμοποιήσει τη μέθοδο hashed sharding στη συλλογή (collection) ***posts***. Χρησιμοποιήσαμε ως index το πεδίο *permalink*. Παρακάτω παρουσιάζουμε πως γίνεται η δημιουργία του hashed index *permalink*,

και πως κάνουμε shard τη posts συλλογή (collection) χρησιμοποιώντας το κλειδί αυτό. Εντός του mongo shell εκτελούμε:
Για να δημιουργήσουμε το index:

```
use blog
db.posts.createIndex( { permalink: "hashed" } )
```

Για να ενεργοποιήσουμε τη δυνατότητα να γίνει sharding στη βάση *blog*:

```
use admin
db.runCommand( { enableSharding: "blog" } )
```

Για να κάνουμε shard τη συλλογή posts:

```
db.runCommand( { shardCollection: "blog.posts", key: { permalink: "hashed" },
unique: false } )
```

Πίνακας 4.7 Αποτελέσματα Εγγραφής Δεδομένων (με Hashed Key “permalink”)

Εγγραφή Δεδομένων		
Εντολή	**Αριθμός χρηστών**	**Διάρκεια**
python test.py --user hpfirstw --data-set-size 400 --prefix a b c d --write-only --parallel 5	10000	0:00:22.777297
python test.py --user hptwow --data-set-size 2000 --prefix a b c d --write-only --parallel 5	50000	0:01:33.108094
python test.py --user hpfivew --data-set-size 5000 --prefix a b c d --write-only --parallel 5	125000	0:03:40.300763

Πίνακας 4.8 Αποτελέσματα Ανάγνωσης Δεδομένων (με Hashed Key “permalink”)

Ανάγνωση Δεδομένων		
Εντολή	**Αριθμός χρηστών**	**Διάρκεια**
python test.py --user hpfirstr --data-set-size 400 --prefix a b c d --read-only --parallel 5	10000	0:00:14.736524
python test.py --user hptwor --data-set-size 2000 --prefix a b c d --read-only --parallel 5	50000	0:01:11.426773
python test.py --user hpfiver --data-set-size 5000 --prefix a b c d --read-only --parallel 5	125000	0:02:49.325215

Πίνακας 4.9 Αποτελέσματα Ανάγνωσης και Εγγραφής Δεδομένων (με Hashed Key “permalink”)

Ανάγνωση και Εγγραφή Δεδομένων		
Εντολή	**Αριθμός χρηστών**	**Διάρκεια**
python test.py --user hpfirst --data-set-size 400 --prefix a b c d --read-write --parallel 5	10000	0:00:30.397645
python test.py --user hptwo --data-set-size 2000 --prefix a b c d --read-write --parallel 5	50000	0:02:04.868303
python test.py --user hpfive --data-set-size 5000 --prefix a b c d --read-write --parallel 5	125000	0:04:58.232668

4.4 Sharded Cluster με ranged-based key compound key (author, permalink)

Σε αυτό το σενάριο, έχουμε χρησιμοποιήσει τη μέθοδο ranged-based sharding στη συλλογή (collection) ***posts***. Χρησιμοποιήσαμε ως index το πεδίο *permalink*. Παρακάτω παρουσιάζουμε πως γίνεται η δημιουργία του index *permalink*, και πως

κάνουμε shard τη posts συλλογή (collection) χρησιμοποιώντας το κλειδί αυτό. Εντός του mongo shell εκτελούμε:

Για να δημιουργήσουμε το index:

```
use blog
db.posts.createIndex( { permalink: 1 } )
```

Για να ενεργοποιήσουμε τη δυνατότητα να γίνει sharding στη βάση *blog*:

```
use admin
db.runCommand( { enableSharding: "blog" } )
```

Για να κάνουμε shard τη συλλογή posts:

```
db.runCommand( { shardCollection: "blog.posts", key: { permalink: 1 } } )
```

Πίνακας 4.10 Αποτελέσματα Εγγραφής Δεδομένων (με Ranged-based key “author”, “permalink”)

Εγγραφή Δεδομένων		
Εντολή	**Αριθμός χρηστών**	**Διάρκεια**
python test.py --user rpfirstw --data-set-size 400 --prefix a b c d --write-only --parallel 5	10000	0:00:22.689622
python test.py --user rptwow --data-set-size 2000 --prefix a b c d --write-only --parallel 5	50000	0:01:31.659743
python test.py --user rpfivew --data-set-size 5000 --prefix a b c d --write-only --parallel 5	125000	0:03:36.301598

Πίνακας 4.11 Αποτελέσματα Ανάγνωσης Δεδομένων (με Ranged-based key “author”, “permalink”)

Ανάγνωση Δεδομένων		
Εντολή	**Αριθμός χρηστών**	**Διάρκεια**
python test.py --user rpfirstr --data-set-size 400 --prefix a b c d --read-only --parallel 5	10000	0:00:14.648495
python test.py --user rptwor --data-set-size 2000 --prefix a b c d --read-only --parallel 5	50000	0:01:12.054163
python test.py --user rpfiver --data-set-size 5000 --prefix a b c d --read-only --parallel 5	125000	0:02:59.922003

Πίνακας 4.12 Αποτελέσματα Ανάγνωσης και Εγγραφής Δεδομένων (με Ranged-based key “author”, “permalink”)

Ανάγνωση και Εγγραφή Δεδομένων		
Εντολή	**Αριθμός χρηστών**	**Διάρκεια**
python test.py --user rpfirst --data-set-size 400 --prefix a b c d --read-write --parallel 5	10000	0:00:30.235428
python test.py --user rptwo --data-set-size 2000 --prefix a b c d --read-write --parallel 5	50000	0:02:06.759618
python test.py --user rpfive --data-set-size 5000 --prefix a b c d --read-write --parallel 5	125000	0:05:04.555127

4.5 Sharded Cluster με ranged-based compound key (permalink, author)

Σε αυτό το σενάριο, έχουμε χρησιμοποιήσει τη μέθοδο ranged-based sharding στη συλλογή (collection) ***posts***. Χρησιμοποιήσαμε ως index ένα συνδιασμό κλειδιών (compound key), χρησιμοποιώντας τα πεδία *permalink* και *author*. Παρακάτω παρουσιάζουμε πως γίνεται η δημιουργία του συνδυασμού κλειδιών,

και πως κάνουμε shard τη posts συλλογή (collection) χρησιμοποιώντας το κλειδί αυτό. Εντός του mongo shell εκτελούμε:

Για να δημιουργήσουμε το index:

```
use blog
db.posts.createIndex( { permalink: 1, author: 1 } )
```

Για να ενεργοποιήσουμε τη δυνατότητα να γίνει sharding στη βάση *blog*:

```
use admin
db.runCommand( { enableSharding: "blog" } )
```

Για να κάνουμε shard τη συλλογή posts:

```
db.runCommand( { shardCollection: "blog.posts", key: { permalink: 1, author:
1 } } )
```

Πίνακας 4.13 Αποτελέσματα Εγγραφής Δεδομένων (με Ranged-based key “permalink”, “author”)

Εγγραφή Δεδομένων		
Εντολή	**Αριθμός χρηστών**	**Διάρκεια**
python test.py --user crfirstw --data-set-size 400 --prefix a b c d --write-only --parallel 5	10000	0:00:22.096260
python test.py --user crtwow --data-set-size 2000 --prefix a b c d --write-only --parallel 5	50000	0:01:32.409863
python test.py --user crfivew --data-set-size 5000 --prefix a b c d --write-only --parallel 5	125000	0:03:35.206866

Πίνακας 4.14 Αποτελέσματα Ανάγνωσης Δεδομένων (με Ranged-based key “permalink”, “author”)

Ανάγνωση Δεδομένων		
Εντολή	**Αριθμός χρηστών**	**Διάρκεια**
python test.py --user crfirstr --data-set-size 400 --prefix a b c d --read-only --parallel 5	10000	0:00:14.988335
python test.py --user crtwor --data-set-size 2000 --prefix a b c d --read-only --parallel 5	50000	0:01:12.383745
python test.py --user crfiver --data-set-size 5000 --prefix a b c d --read-only --parallel 5	125000	0:02:58.113081

Πίνακας 4.15 Αποτελέσματα Ανάγνωσης και Εγγραφής Δεδομένων (με Ranged-based key “permalink”, “author”)

Ανάγνωση και Εγγραφή Δεδομένων		
Εντολή	**Αριθμός χρηστών**	**Διάρκεια**
python test.py --user crfirst --data-set-size 400 --prefix a b c d --read-write	10000	0:00:31.403356
python test.py --user crtwo --data-set-size 2000 --prefix a b c d --read-write	50000	0:02:06.618233
python test.py --user crfive --data-set-size 5000 --prefix a b c d --read-write --parallel 5	125000	0:05:03.712846

Στον παρακάτω πίνακα (Πίνακας 4.16), γίνεται σύγκριση των μετρήσεων που πήραμε, με τη κάθε μέθοδο sharding. Περιλαμβάνεται η διάρκεια εκτέλεσης των tests για 125,000 χρήστες αποκλειστικά.

Πίνακας 4.16 Σύγκριση αποτελεσμάτων

Είδος Sharding	Διάρκεια Εγραφής Δεδομένων	Διάρκεια Ανάγνωσης Δεδομένων	Διάρκεια Ανάγνωσης και Εγραφής Δεδομένων
Sharded Cluster με αρχικές ρυθμίσεις	0:12:03.552131	0:19:00.162130	0:21:25.374162
Sharded Cluster με hashed key το _id	0:12:32.910953	0:19:39.380095	0:21:40.494806
Sharded Cluster με hashed key το permalink	0:03:40.300763	0:02:49.325215	0:04:58.232668
Sharded Cluster με ranged-based key (permalink)	0:03:36.301598	0:02:59.922003	0:05:04.555127
Sharded Cluster με ranged-based key compound key (permalink, author)	0:03:35.206866	0:02:58.113081	0:05:03.712846

ΚΕΦΑΛΑΙΟ 5

Στην παρούσα ενότητα παρουσιάζουμε τα συμπεράσματα της πτυχιακής και παράλληλα, γίνεται μια σύνοψη των μελλοντικών επεκτάσεων που επιθυμούμε να πραγματοποιηθούν.

5. ΕΠΙΛΟΓΟΣ

5.1 Σύνοψη και συμπεράσματα

Στόχος της συγκεκριμένης πτυχιακής ήταν να δημιουργήσουμε ένα κατανεμημένο σύστημα, το οποίο και καταφέραμε με τη βοήθεια της τεχνολογίας sharding της MongoDB. Με τη βοήθεια του Πίνακα 4.16, μπορούμε να συγκρίνουμε τις διάφορες στρατηγικές sharding και τα διάφορα shard κλειδιά που χρησιμοποιήσαμε και να επιλέξουμε ποιο είναι το πιο σωστό για τη δική μας εφαρμογή.

Όπως παρατηρούμε, οι δύο πρώτες γραμμές έχουν τους πιο μεγάλους χρόνους. Στη πρώτη γραμμή, έχουμε τις τιμές για Sharded Cluster με αρχικές ρυθμίσεις, στο οποιο δεν έχει γίνει shard κάποια συλλογή (collection), οπότε όλες οι συλλογές (collections) της εφαρμογής μας θα τοποθετηθούν στο πρωτεύων (primary) shard. Έτσι, όλες οι αιτήσεις, θα πηγαίνουν μόνο σε ένα μηχάνημα. Στη δεύτερη περίπτωση, έχουμε κάνει shard τη συλλογή ***posts***, με το hashed shard κλειδί *_id*. Τα δεδομένα μας σε αυτή τη περίπτωση δε βρίσκονται μόνο στο πρωτεύων shard, αλλά και στα δευτερεύοντα (secondary). Αλλά, από τη στιγμή που καμία από τις αιτήσεις (queries) μας δε συμπεριλαμβάνει το κλειδί *_id*, η MongoDB αναγκάζεται να στείλει τις αιτήσεις σε όλα τα shards (broadcast), το οποίο, στη προκειμένη περίπτωση, είναι πιο χρονοβόρο σε σχέση με τη πρώτη περίπτωση, που δεν είχαμε καμία sharded συλλογή, και όλα τα δεδομένα ήταν στο πρωτεύων shard.

Από την άλλη, βλέπουμε στις γραμμές τρία, τέσσερα και πέντε, ότι ο χρόνος εκτέλεσης μειώνεται αισθητά. Ο λόγος είναι ότι και στις τρεις περιπτώσεις χρησιμοποιείται ως shard κλειδί (είτε είναι μέρος του shard κλειδιού) το *permalink*, που χρησιμοποιείται στις περισσότερες αιτήσεις μας.

Από τη στιγμή που οι γραμμές τέσσερα και πέντε χρησιμοποιούν την τεχνική ranged-based sharding, έχουν παρόμοια αποτελέσματα, διότι για την ανάγνωση ενός άρθρου, χρησιμοποιούμε μόνο το *permalink*.

Τα πιο σημαντικά αποτελέσματα, είναι αυτά της γραμμής τρία. Όπως παρατηρούμε ο χρόνος ανάγνωσης είναι αισθητά πιο μικρός από κάθε άλλη μέθοδο/συνδιασμό κλειδιών και παράλληλα ο χρόνος εγγραφής, λίγο πιο αργός σε σχέση με τη τεχνική ranged-based sharding.

Από τη στιγμή, που σε ένα blog, πιο συχνά γίνονται αναγνώσεις παρά εγγραφές, το hashed shard κλειδί *permalink*, είναι το ιδανικότερο για τη δική μας περίπτωση.

5.2 Μελλοντικές επεκτάσεις

Τη στιγμή που υλοποιήθηκε η συγκεκριμένη πτυχιακή εργασία, παραδίδουμε μια υλοποίηση ενός web blog, το οποίο υποστηρίζει οριζόντια επεκτασημότητα και είναι έτοιμο να εγκατασταθεί και να χρησιμοποιηθεί άμεσα. Όσον αφορά τις μελλοντικές επεκτάσεις, θα θέλαμε να γίνει μια αναθεώρηση του front-end του web blog και να εμβαθύνουμε στις μεθόδους ασφαλείας του συστήματος. Τέλος θα θέλαμε να χρησιμοποιήσουμε τεχνολογίες για διαχείριση containers όπως Kubernetes, για να εξασφαλίσουμε την ομαλή λειτουργία και επεκτασιμότητα του συστήματος σε περιπτώσεις υψηλού φόρτου.

ΚΕΦΑΛΑΙΟ 6

Στην παρούσα ενότητα περιλαμβάνονται όλοι οι πηγαίοι κώδικες και τα scripts του συστήματος.

6. ΠΑΡΑΡΤΗΜΑ ΚΩΔΙΚΑ

6.1 Java Εφαρμογή

6.1.1 Java κλάσεις

BlogController.java

```
package blog;



import java.io.IOException;
import java.io.StringWriter;
import java.io.Writer;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import javax.servlet.http.Cookie;
import com.mongodb.MongoClient;
import com.mongodb.MongoClientURI;
import com.mongodb.client.MongoDatabase;
import freemarker.template.Configuration;
import freemarker.template.SimpleHash;
import freemarker.template.Template;
import freemarker.template.TemplateException;
import org.apache.commons.configuration.ConfigurationException;
import org.apache.commons.configuration.PropertiesConfiguration;
import org.apache.commons.lang3.StringEscapeUtils;
import org.bson.Document;
import spark.Request;
```

```
import spark.Response;
import spark.Route;

import static spark.Spark.get;
import static spark.Spark.post;
import static spark.Spark.setPort;

/**
 * This class encapsulates the controllers for the blog web application.  It delegates all interaction
with MongoDB
 * to three Data Access Objects (DAOs).
 * <p/>
 * It is also the entry point into the web application.
 */
public class BlogController {
        private final Configuration cfg;
        private final BlogPostDAO blogPostDAO;
        private final UserDAO userDAO;
        private final SessionDAO sessionDAO;

        public static void main(String[] args) throws IOException {
            try
            {
                PropertiesConfiguration config = new
PropertiesConfiguration(
                        "src/main/resources/mongodb.properties");

                String bindIp = config.getString("bindIp");
                String port = config.getString("port");

                if (args.length == 0) {
                    new BlogController("mongodb://" + bindIp + ":" + port);
                }
                else {
                    new BlogController(args[0]);
                }
            }
```

```
            catch (ConfigurationException cex) {
            }
        }

        public BlogController(String mongoURIString) throws IOException {
            final MongoClient mongoClient = new MongoClient(new
MongoClientURI(mongoURIString));
            final MongoDatabase blogDatabase =
mongoClient.getDatabase("blog");

            blogPostDAO = new BlogPostDAO(blogDatabase);
            userDAO = new UserDAO(blogDatabase);
            sessionDAO = new SessionDAO(blogDatabase);

            cfg = createFreemarkerConfiguration();
            setPort(8082);
            initializeRoutes();
        }

        abstract  class FreemarkerBasedRoute extends Route {
            final Template template;

            /**
        * Constructor
        *
        * @param path The route path which is used for matching. (e.g. /hello, users/:name)
        */
            protected FreemarkerBasedRoute(final String path, final String templateName)
            throws IOException {
                super(path);
                template = cfg.getTemplate(templateName);
            }

            @Override
            public Object handle(Request request, Response response) {
                StringWriter writer = new StringWriter();
                try {
```

```
                    doHandle(request, response, writer);
                } catch (Exception e) {
                    e.printStackTrace();
                    response.redirect("/internal_error");
                }
                return writer;
            }

            protected abstract void doHandle(
            final Request request, final Response response, final Writer writer)
            throws IOException, TemplateException;

        }

        private void initializeRoutes() throws IOException {
            // this is the blog home page
            get(new FreemarkerBasedRoute("/", "blog_template.ftl") {
                @Override
                public void doHandle(Request request, Response response, Writer writer)
                throws IOException, TemplateException {
                    String username =
sessionDAO.findUserNameBySessionId(getSessionCookie(request));

                    List<Document> posts =
blogPostDAO.findByDateDescending(10);
                    SimpleHash root = new SimpleHash();

                    root.put("myposts", posts);
                    root.put("indexOfOldEntries", 2);
                    if (username != null && !username.isEmpty()) {
                        root.put("username", username);
                    }

                    template.process(root, writer);
                }
            });
```

```
            // used to display blog posts
            get(new FreemarkerBasedRoute("/page/:pagenum", "posts_template.ftl") {
                @Override
                protected void doHandle(Request request, Response response, Writer writer)
            throws IOException, TemplateException {
                    int pagenum = Integer.parseInt(
                            (request.params(":pagenum")));
                    String username =
sessionDAO.findUserNameBySessionId(getSessionCookie(request));

                    if (pagenum > 0) {
                        SimpleHash root = new SimpleHash();

                        if (pagenum == 1) {
                            response.redirect("/");
                        } else {
                            List<Document> posts =

blogPostDAO.findByDateDescendingIndex((pagenum * 10) - 9
                                                    -1, 10);
                            root.put("myposts", posts);
                        }

                        root.put("indexOfOldEntries", pagenum + 1);
                        root.put("indexOfNewEntries", pagenum - 1);

                        if (username != null && !username.isEmpty()) {
                            root.put("username", username);
                        }
                        template.process(root, writer);
                    }
                    else {
                        response.redirect("/page_not_found");
                    }
                }
            });
```

```
            // used to display actual blog post detail page
            get(new FreemarkerBasedRoute("/post/:permalink", "entry_template.ftl") {
                @Override
                protected void doHandle(Request request, Response response, Writer writer)
            throws IOException, TemplateException {
                    String username =
sessionDAO.findUserNameBySessionId(getSessionCookie(request));
                    String permalink = request.params(":permalink");

                    Document post =
blogPostDAO.findByPermalink(permalink);
                    if (post == null) {
                        response.redirect("/post_not_found");
                    }
                    else {
                        // empty comment to hold new comment in form at bottom of blog
entry detail page
                        SimpleHash newComment = new SimpleHash();
                        newComment.put("name", "");
                        newComment.put("email", "");
                        newComment.put("body", "");

                        SimpleHash root = new SimpleHash();

                        root.put("post", post);
                        root.put("comment", newComment);

                        if (username != null) {
                            root.put("username", username);
                        }

                        template.process(root, writer);
                    }
                }
            });

            // handle the signup post
```

```
        post(new FreemarkerBasedRoute("/signup", "signup.ftl") {
            @Override
            protected void doHandle(Request request, Response response, Writer writer)
            throws IOException, TemplateException {
                String firstName = request.queryParams("firstname");
                String lastName = request.queryParams("lastname");
                String username = request.queryParams("username");
                String email = request.queryParams("email");
                String password = request.queryParams("password");
                String verify = request.queryParams("verify");

                HashMap<String, String> root = new HashMap<String,
String>();
                root.put("firstName",
StringEscapeUtils.escapeHtml4(firstName));
                root.put("lastName",
StringEscapeUtils.escapeHtml4(lastName));
                root.put("username",
StringEscapeUtils.escapeHtml4(username));
                root.put("email",
StringEscapeUtils.escapeHtml4(email));

                if (validateSignup(firstName, lastName, username,
password, verify, email, root)) {
                    // creating user
                    if (!userDAO.addUser(firstName, lastName,
username, password, email)) {
                        // duplicate user
                        root.put("username_error", "Username already in use,
Please choose another");
                        template.process(root, writer);
                    }
                    else {
                        // good user, let's start a session
                        String sessionID =
sessionDAO.startSession(username);
```

```
                        response.raw().addCookie(new Cookie("session",
sessionID));
                        response.redirect("/welcome");
                    }
                }
                else {
                    // bad signup
                    template.process(root, writer);
                }
            }
        });

        // present signup form for blog
        get(new FreemarkerBasedRoute("/signup", "signup.ftl") {
            @Override
            protected void doHandle(Request request, Response response, Writer writer)
         throws IOException, TemplateException {

                SimpleHash root = new SimpleHash();

                // initialize values for the form.
                root.put("firstName", "");
                root.put("lastName", "");
                root.put("username", "");
                root.put("password", "");
                root.put("email", "");
                root.put("firstNamed_error", "");
                root.put("lastNamed_error", "");
                root.put("username_error", "");
                root.put("email_error", "");
                root.put("password_error", "");
                root.put("verify_error", "");

                template.process(root, writer);
            }
        });
```

```
            // will present the form used to process new blog posts
            get(new FreemarkerBasedRoute("/newpost", "newpost_template.ftl") {
                @Override
                protected void doHandle(Request request, Response response, Writer writer)
                throws IOException, TemplateException {
                    // get cookie
                    String username =
sessionDAO.findUserNameBySessionId(getSessionCookie(request));

                    if (username == null) {
                        // looks like a bad request. user is not logged in
                        response.redirect("/login");
                    }
                    else {
                        SimpleHash root = new SimpleHash();
                        root.put("username", username);

                        template.process(root, writer);
                    }
                }
            });

            // handle the new post submission
            post(new FreemarkerBasedRoute("/newpost", "newpost_template.ftl") {
                @Override
                protected void doHandle(Request request, Response response, Writer writer)
                throws IOException, TemplateException {
                    String title =
StringEscapeUtils.escapeHtml4(request.queryParams("subject"));
                    String post =
StringEscapeUtils.escapeHtml4(request.queryParams("body"));
                    String tags =
StringEscapeUtils.escapeHtml4(request.queryParams("tags"));
                    String username =
sessionDAO.findUserNameBySessionId(getSessionCookie(request));

                    if (username == null) {
```

```
                        response.redirect("/login");        // only logged in users can
post to blog
                    }
                    else if (title.equals("") || post.equals("")) {
                        // redisplay page with errors
                        HashMap<String, String> root = new
HashMap<String, String>();
                        root.put("errors", "post must contain a title and blog entry.");
                        root.put("subject", title);
                        root.put("username", username);
                        root.put("tags", tags);
                        root.put("body", post);
                        template.process(root, writer);
                    }
                    else {
                        // extract tags
                        ArrayList<String> tagsArray =
extractTags(tags);

                        // substitute some <p> for the paragraph breaks
                        post = post.replaceAll("\\r?\\n", "<p>");

                        String permalink = blogPostDAO.addPost(title,
post, tagsArray, username);
                        // now redirect to the blog permalink
                        response.redirect("/post/" + permalink);
                    }
                }
            });

            get(new FreemarkerBasedRoute("/welcome", "welcome.ftl") {
                @Override
                protected void doHandle(Request request, Response response, Writer writer)
             throws IOException, TemplateException {
                    String cookie = getSessionCookie(request);
                    String username =
sessionDAO.findUserNameBySessionId(cookie);
```

```
                Map<String, String> userDetails =
userDAO.getUserDetails(username);

                if (username == null) {
                    // can't identify the user, redirecting to signup
                    response.redirect("/signup");
                }
                else {
                    SimpleHash root = new SimpleHash();
                    root.put("firstName", userDetails.get("firstName"));
                    template.process(root, writer);
                }
            }
        });

        // process a new comment
        post(new FreemarkerBasedRoute("/newcomment", "entry_template.ftl") {
            @Override
            protected void doHandle(Request request, Response response, Writer writer)
            throws IOException, TemplateException {
                String cookie = getSessionCookie(request);
                String username =
sessionDAO.findUserNameBySessionId(cookie);
                Map<String, String> userDetails =
userDAO.getUserDetails(username);
                String name;
                String email;

                if (username == null) {
                    name =
StringEscapeUtils.escapeHtml4(request.queryParams("commentName"));
                    email =
StringEscapeUtils.escapeHtml4(request.queryParams("commentEmail"));
                }
                else {
                    name = String.format("%s %s",
```

```
                                userDetails.get("firstName"),
userDetails.get("lastName"));
                        email = userDetails.get("email");
                    }

                    String body =
StringEscapeUtils.escapeHtml4(request.queryParams("commentBody"));
                    String permalink = request.queryParams("permalink");
                    Document post =
blogPostDAO.findByPermalink(permalink);

                    if (post == null) {
                        response.redirect("/post_not_found");
                    }
                    // check that comment is good
                    else if (name.equals("") || body.equals("")) {
                        // bounce this back to the user for correction
                        SimpleHash root = new SimpleHash();
                        SimpleHash comment = new SimpleHash();

                        comment.put("name", name);
                        comment.put("email", email);
                        comment.put("body", body);
                        root.put("comment", comment);
                        root.put("post", post);
                        root.put("errors", "Post must contain your name and an actual
comment");

                        template.process(root, writer);
                    }
                    else {
                        blogPostDAO.addPostComment(name,
                                email, body, post.getString("author"),
permalink);
                        response.redirect("/post/" + permalink);
                    }
                }
```

```
        });

        // present the login page
        get(new FreemarkerBasedRoute("/login", "login.ftl") {
            @Override
            protected void doHandle(Request request, Response response, Writer writer)
            throws IOException, TemplateException {
                SimpleHash root = new SimpleHash();

                root.put("username", "");
                root.put("login_error", "");

                template.process(root, writer);
            }
        });

        // process output coming from login form. On success redirect folks to the welcome page
        // on failure, just return an error and let them try again.
        post(new FreemarkerBasedRoute("/login", "login.ftl") {
            @Override
            protected void doHandle(Request request, Response response, Writer writer)
            throws IOException, TemplateException {
                String username = request.queryParams("username");
                String password = request.queryParams("password");
                Document user = userDAO.validateLogin(username,
password);

                if (user != null) {
                    // valid user, let's log them in
                    String sessionID =
sessionDAO.startSession(String.valueOf(user.get("_id")));

                    if (sessionID == null) {
                        response.redirect("/internal_error");
                    }
                    else {
                        // set the cookie for the user's browser
```

```
                        response.raw().addCookie(new Cookie("session",
sessionID));
                        response.redirect("/welcome");
                    }
                }
                else {
                    SimpleHash root = new SimpleHash();

                    root.put("username",
StringEscapeUtils.escapeHtml4(username));
                    root.put("password", "");
                    root.put("login_error", "Invalid Login");
                    template.process(root, writer);
                }
            }
        });

        // Show the posts filed under a certain tag
        get(new FreemarkerBasedRoute("/tag/:thetag", "blog_template.ftl") {
            @Override
            protected void doHandle(Request request, Response response, Writer writer)
            throws IOException, TemplateException {

                String username =
sessionDAO.findUserNameBySessionId(getSessionCookie(request));
                SimpleHash root = new SimpleHash();
                String tag =
StringEscapeUtils.escapeHtml4(request.params(":thetag"));
                List<Document> posts =
blogPostDAO.findByTagDateDescending(tag);

                root.put("myposts", posts);
                if (username != null) {
                    root.put("username", username);
                }
                template.process(root, writer);
            }
```

```
            });

            // will allow a user to click Like on a post
            post(new FreemarkerBasedRoute("/like", "entry_template.ftl") {
                @Override
                protected void doHandle(Request request, Response response, Writer writer)
                throws IOException, TemplateException {
                    String permalink = request.queryParams("permalink");
                    String commentOrdinalStr =
request.queryParams("comment_ordinal");

                    // look up the post in question
                    int ordinal = Integer.parseInt(commentOrdinalStr);

                    String username =
sessionDAO.findUserNameBySessionId(getSessionCookie(request));
                    if (username == null) {
                        response.redirect("/login");       // only logged in users can
Like a post
                    }
                    else {
                        Document post =
blogPostDAO.findByPermalink(permalink);

                        //  if post not found, redirect to post not found error
                        if (post == null) {
                            response.redirect("/post_not_found");
                        }
                        else {
                            blogPostDAO.likePost(permalink, ordinal);
                            response.redirect("/post/" + permalink);
                        }
                    }
                }
            });

            // tells the user that the URL is dead
```

```
        get(new FreemarkerBasedRoute("/post_not_found", "post_not_found.ftl")
{
            @Override
            protected void doHandle(Request request, Response response, Writer writer)
        throws IOException, TemplateException {
                SimpleHash root = new SimpleHash();
                template.process(root, writer);
            }
        });

        // allows the user to logout of the blog
        get(new FreemarkerBasedRoute("/logout", "signup.ftl") {
            @Override
            protected void doHandle(Request request, Response response, Writer writer)
        throws IOException, TemplateException {
                String sessionID = getSessionCookie(request);

                if (sessionID == null) {
                    // no session to end
                    response.redirect("/login");
                }
                else {
                    // deletes from session table
                    sessionDAO.endSession(sessionID);

                    // this should delete the cookie
                    Cookie c = getSessionCookieActual(request);
                    c.setMaxAge(0);

                    response.raw().addCookie(c);
                    response.redirect("/login");
                }
            }
        });

        // will present the 'About me' section
        get(new FreemarkerBasedRoute("/about", "about.ftl") {
```

```
                @Override
                protected void doHandle(Request request, Response response, Writer writer)
            throws IOException, TemplateException {
                    String username =
sessionDAO.findUserNameBySessionId(getSessionCookie(request));
                    SimpleHash root = new SimpleHash();

                    if (username != null) {
                        root.put("username", username);
                    }
                    template.process(root, writer);
                }
            });

            // will present the form used to submit a question/report
            get(new FreemarkerBasedRoute("/contact", "contact.ftl") {
                @Override
                protected void doHandle(Request request, Response response, Writer writer)
            throws IOException, TemplateException {
                    String username =
sessionDAO.findUserNameBySessionId(getSessionCookie(request));
                    SimpleHash root = new SimpleHash();

                    if (username != null) {
                        root.put("username", username);
                    }
                    template.process(root, writer);
                }
            });

            // used to process page not found error
            get(new FreemarkerBasedRoute("/page_not_found", "page_not_found.ftl")
{
                @Override
                protected void doHandle(Request request, Response response, Writer writer)
            throws IOException, TemplateException {
                    SimpleHash root = new SimpleHash();
```

```
                root.put("error", "Page not found.");
                template.process(root, writer);
            }
        });

        // used to process internal errors
        get(new FreemarkerBasedRoute("/internal_error", "error_template.ftl") {
            @Override
            protected void doHandle(Request request, Response response, Writer writer)
            throws IOException, TemplateException {
                SimpleHash root = new SimpleHash();

                root.put("error", "System has encountered an error.");
                template.process(root, writer);
            }
        });
    }

    // helper function to get session cookie as string
    private String getSessionCookie(final Request request) {
        if (request.raw().getCookies() == null) {
            return null;
        }
        for (Cookie cookie : request.raw().getCookies()) {
            if (cookie.getName().equals("session")) {
                return cookie.getValue();
            }
        }
        return null;
    }

    // helper function to get session cookie as string
    private Cookie getSessionCookieActual(final Request request) {
        if (request.raw().getCookies() == null) {
            return null;
        }
```

```
        for (Cookie cookie : request.raw().getCookies()) {
            if (cookie.getName().equals("session")) {
                return cookie;
            }
        }
        return null;
    }

    // tags the tags string and put it into an array
    private ArrayList<String> extractTags(String tags) {
        // probably more efficent ways to do this.
        //
        // whitespace = re.compile('\s')
        String tagArray[] = tags.replaceAll("\\s", "")
                .split(",");

        // let's clean it up, removing the empty string and removing dups
        ArrayList<String> cleaned = new ArrayList<String>();
        for (String tag : tagArray) {
            if (!tag.equals("") && !cleaned.contains(tag)) {
                cleaned.add(tag);
            }
        }
        return cleaned;
    }

    // validates that the registration form has been filled out right and username conforms
    public boolean validateSignup(String firstName,
                  String lastName,
                  String username,
                  String password,
                  String verify,
                  String email,
                  HashMap<String, String> errors) {
        String LETTERS_RE = "^[a-zA-Z]{3,20}$";
        String USER_RE = "^[a-zA-Z0-9_-]{3,20}$";
        String PASS_RE = "^.{3,20}$";
```

```
        String EMAIL_RE = "^[\\S]+@[\\S]+\\.[\\S]+$";

        errors.put("firstName_error", "");
        errors.put("lastName_error", "");
        errors.put("username_error", "");
        errors.put("password_error", "");
        errors.put("verify_error", "");
        errors.put("email_error", "");

        if (!firstName.matches(LETTERS_RE)) {
            errors.put("firstName_error", "Please use only letters (a-z)");
            return false;
        }

        if (!lastName.matches(LETTERS_RE)) {
            errors.put("lastName_error", "Please use only letters (a-z)");
            return false;
        }

        if (!username.matches(USER_RE)) {
            errors.put("username_error", "Please use only letters (a-z), numbers");
            return false;
        }

        if (!password.matches(PASS_RE)) {
            errors.put("password_error", "Invalid password.");
            return false;
        }

        if (!password.equals(verify)) {
            errors.put("verify_error", "These passwords don't match. Try again?");
            return false;
        }

        if (!email.equals("")) {
            if (!email.matches(EMAIL_RE)) {
                errors.put("email_error", "Please enter a valid email address");
```

```
                    return false;
                }
            }
            return true;
        }

        private Configuration createFreemarkerConfiguration() {
            Configuration retVal = new Configuration();
            retVal.setClassForTemplateLoading(BlogController.class,
"/freemarker");
            return retVal;
        }
    }
```

BlogPostDAO.java

```
package blog;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;

import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import org.bson.Document;

import static com.mongodb.client.model.Filters.and;
import static com.mongodb.client.model.Filters.eq;

public class BlogPostDAO {
    MongoCollection<Document> postsCollection;

    public BlogPostDAO(final MongoDatabase blogDatabase) {
        postsCollection = blogDatabase.getCollection("posts");
    }
```

```
        public Document findByPermalink(String permalink) {
            Document post = postsCollection.find(eq("permalink",
permalink)).first();

              // fix up if a post has no likes
            if (post != null) {
                List<Document> comments = (List<Document>)
post.get("comments");
                for (Document comment : comments) {
                    if (!comment.containsKey("num_likes")) {
                        comment.put("num_likes", 0);
                    }
                }
            }
            return post;
        }

        public List<Document> findByDateDescending(int limit) {
            List<Document> posts = postsCollection.find()
                    .sort(new Document("date", -1))
                    .limit(limit)
                    .into(new ArrayList<Document>());
            return posts;
        }

        public List<Document> findByDateDescendingIndex(int index, int limit) {
            List<Document> posts = postsCollection.find()
                    .sort(new Document("date", -1))
                    .skip(index)
                    .limit(limit)
                    .into(new ArrayList<Document>());
            return posts;
        }

        public List<Document> findByTagDateDescending(final String tag) {
            List<Document> posts = postsCollection.find(eq("tags", tag))
```

```
                .sort(new Document("date", -1))
                .limit(10)
                .into(new ArrayList<Document>());
        return posts;
    }

    public String addPost(String title, String body, List tags, String username) {
        // whitespace becomes _, get rid of non alphanumeric
        String permalink = title.replaceAll("\\s", "_")
                .replaceAll("\\W", "")
                .toLowerCase();

        Document post = new Document("title", title);
        post.append("author", username);
        post.append("body", body);
        post.append("permalink", permalink);
        post.append("tags", tags);
        post.append("comments", new java.util.ArrayList());
        post.append("date", new java.util.Date());

        try {
            postsCollection.insertOne(post);
        } catch (Exception e) {
            return null;
        }
        return permalink;
    }

    public void addPostComment(final String name,
                               final String email,
                               final String body,
                               final String author,
                               final String permalink) {
        Document comment = new Document("author", name)
                .append("body", body);

        if (email != null && !email.isEmpty()) {
```

```
                comment.append("email", email);
            }

            postsCollection.updateOne(
                    eq("permalink", permalink),
                    new Document("$push",
                            new Document("comments", comment)));
        }

        public void likePost( final String permalink,  final int ordinal) {
            postsCollection.updateOne(
                    eq("permalink", permalink),
                    new Document("$inc", new Document("comments." + ordinal
+ ".num_likes", 1)));
        }
    }
```

SessionDAO.java

```
package blog;

import java.security.SecureRandom;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import org.bson.Document;
import sun.misc.BASE64Encoder;

import static com.mongodb.client.model.Filters.eq;

public class SessionDAO {
    private final MongoCollection<Document> sessionsCollection;

    public SessionDAO(final MongoDatabase blogDatabase) {
        sessionsCollection = blogDatabase.getCollection("sessions");
    }
```

```
        public String findUserNameBySessionId(String sessionId) {
            Document session = getSession(sessionId);
            return (session == null) ? null :
String.valueOf(session.get("username"));
        }

        // starts a new session in the sessions table
        public String startSession(String username) {

            // get 32 byte random number. that's a lot of bits.
            SecureRandom generator = new SecureRandom();
            byte randomBytes[] = new byte[32];
            generator.nextBytes(randomBytes);

            BASE64Encoder encoder = new BASE64Encoder();

            String sessionID = encoder.encode(randomBytes);

            // build the BSON object
            Document session = new Document("username", username);

            session.append("_id", sessionID);

            sessionsCollection.deleteMany(
                    new Document("username",username));

            sessionsCollection.insertOne(session);

            return session.getString("_id");
        }

        // ends the session by deleting it from the sessions table
        public void endSession(String sessionID) {
            sessionsCollection.deleteOne(eq("_id", sessionID));
        }
```

```
        // retrieves the session from the sessions table
        public Document getSession(String sessionID) {
            return sessionsCollection.find(eq("_id", sessionID)).first();
        }
}
```

UserDAO.java

```
package blog;

import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;
import java.util.HashMap;
import java.util.Map;
import java.util.Random;
import com.mongodb.MongoWriteException;
import com.mongodb.client.FindIterable;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoCursor;
import com.mongodb.client.MongoDatabase;
import org.bson.Document;
import sun.misc.BASE64Encoder;

import static com.mongodb.client.model.Filters.eq;

public class UserDAO {
    private final MongoCollection<Document> usersCollection;
    private Random random = new SecureRandom();

    public UserDAO(final MongoDatabase blogDatabase) {
        usersCollection = blogDatabase.getCollection("users");
    }

    // validates that username is unique and insert into db
    public boolean addUser(String firstName,
```

```
                String lastName,
                String username,
                String password,
                String email) {

        String passwordHash = makePasswordHash(password,
String.valueOf(random.nextInt()));

        Document user = new Document();

        user.append("_id", username)
                .append("firstName", firstName)
                .append("lastName", lastName)
                .append("password", passwordHash);

        if (email != null && !email.equals("")) {
            // the provided email address
            user.append("email", email);
        }

        try {
            usersCollection.insertOne(user);
            return true;
        } catch (MongoWriteException e) {
            return false;
        }
    }

    public Document validateLogin(String username, String password) {
        Document user = usersCollection.find(eq("_id",
username)).first();

        if (user == null) {
            return null;
        }
```

```
            String hashedAndSalted =
String.valueOf(user.get("password"));

            String salt = hashedAndSalted.split(",")[1];

            if (!hashedAndSalted.equals(makePasswordHash(password,
salt))) {
                return null;
            }
            return user;
        }

        // retrieves user details
        public Map<String, String> getUserDetails(String username) {
            Document query = new Document("_id", username);
            FindIterable<Document> find = usersCollection.find(query);
            Map<String, String> userDetails = new HashMap<String,
String>();

            MongoCursor<Document> cursor = find.iterator();
            try {
                while (cursor.hasNext()) {
                    Document doc = cursor.next();
                    userDetails.put("firstName", doc.getString("firstName"));
                    userDetails.put("lastName", doc.getString("lastName"));
                    userDetails.put("email", doc.getString("email"));
                }
            } finally {
                cursor.close();
            }

            return userDetails;
        }

        private String makePasswordHash(String password, String salt) {
            try {
                String saltedAndHashed = password + "," + salt;
```

```
                MessageDigest digest =
MessageDigest.getInstance("MD5");
                digest.update(saltedAndHashed.getBytes());
                BASE64Encoder encoder = new BASE64Encoder();
                byte hashedBytes[] = (new String(digest.digest(), "UTF-
8")).getBytes();
                return encoder.encode(hashedBytes) + "," + salt;
            } catch (NoSuchAlgorithmException e) {
                throw new RuntimeException("MD5 is not available", e);
            } catch (UnsupportedEncodingException e) {
                throw new RuntimeException("UTF-8 unavailable? Not a chance", e);
            }
        }
    }
```

6.1.2 FreeMarker Templates

about.ftl

```
<!DOCTYPE html>
<html>
<head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>About</title>
          <!--http://osdir.com/ml/web.freemarker.user/2005-08/msg00004.html-->
        <style type="text/css">
        <#include "styles.css">
    </style>
</head>

<body>
        <div id=header>
                <h1>My Blog</h1>
        </div>
        <nav>
                <ul id="top-nav">
```

```
                <li><a href="/">HOME</a></li>
                <li><a href="/about">ABOUT</a></li>
                <li><a href="/contact">CONTACT</a></li>
                <#if username??>
                    <ul style="float:right;list-style-type:none;">
                        <li><a href="/newpost">NEW POST</a></li>
                        <li><a href="/logout">LOGOUT</a></li>
                    </ul>
                <#else>
                    <ul style="float:right;list-style-type:none;">
                        <li><a href="/login">LOGIN</a></li>
                        <li><a href="/signup">SIGN UP</a></li>
                    </ul>
                </#if>
            </ul>
        </nav>
        <div id="about">
            <h2>About</h2>
            <p>
  This project is part of a BSc thesis (Development of a Distributed System using Cloud Services), developed by Alexandros Efthymiadis and Ioannis Chaldaios.</p>

            <p>You can find our source code on <a href="http://www.github.com/">Github</a></p>
        </div>
        <div class="spacer"></div>
        <footer>
            <h2>Copyright &copy; 2016 Alexandros Efthymiadis - Ioannis Chaldaios B· All rights reserved</h2>
        </footer>
    </body>
    </html>
```

blog_template.ftl

```
<!DOCTYPE html>
<html>
<head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>My Blog</title>
          <!--http://osdir.com/ml/web.freemarker.user/2005-08/msg00004.html-->
        <style type="text/css">
        <#include "styles.css">
    </style>
        <script type="text/javascript">var switchTo5x=true;</script>
        <script type="text/javascript" src="http://w.sharethis.com/button/buttons.js"></script>
        <script type="text/javascript">stLight.options({publisher: "f9508b6a-a83d-4738-bb15-
37e27d5120f0", doNotHash: false, doNotCopy: false, hashAddressBar: false});</script>

</head>

<body>
        <div id=header>
                <h1>My Blog</h1>
        </div>
        <nav>
                <ul id="top-nav">
                        <li><a href="/">HOME</a></li>
                        <li><a href="/about">ABOUT</a></li>
                        <li><a href="/contact">CONTACT</a></li>
                        <#if username??>
                                <ul style="float:right;list-style-type:none;">
                                        <li><a href="/newpost">NEW POST</a></li>
                                        <li><a href="/logout">LOGOUT</a></li>
                                </ul>
                        <#else>
                                <ul style="float:right;list-style-type:none;">
                                        <li><a href="/login">LOGIN</a></li>
                                        <li><a href="/signup">SIGN UP</a></li>
                                </ul>
                        </#if>
                </ul>
```

```
        </nav>
        <div id="container">
            <div id="sidebar">
                <h2>SHARE</h2>
                <div class="navlist">
                    <ul>
                        <li><span class='st_facebook_large'
displayText='Facebook'></span></li>
                        <li><span class='st_twitter_large' displayText='Tweet'></span></li>
                        <li><span class='st_googleplus_large' displayText='Google
+'></span></li>
                        <li><span class='st_linkedin_large'
displayText='LinkedIn'></span></li>
                        <li><span class='st_email_large' displayText='Email'></span></li>
                    </ul>
                </div>
            </div>



            <#list myposts as post>
                <div id="title">
                    <h2><a
href="/post/${post["permalink"]}">${post["title"]}</a></h2>
                </div>
                <div id="info">
                    <b>Posted</b> ${post["date"]?datetime} | <b>By</b>
${post["author"]}<br>
                </div>
                <br>
                <div id="content">
                    ${post["body"]!""}
                </div>
                <div id=tags>
                    Tags:
                    <#if post["tags"]??>
                        <#list post["tags"] as tag>
```

```
                                <a href="/tag/${tag}">${tag}</a>
                            </#list>
                        </#if>
                        |
                        Comments:
                        <#if post["comments"]??>
                            <#assign numComments = post["comments"]?size>
                                <#else>
                                    <#assign numComments = 0>
                        </#if>
                        <a href="/post/${post["permalink"]}">${numComments}</a>
                    </div>
                </#list>
            </div>
            <#if indexOfOldEntries??>
                <div class="pagination">
                    <ul>
                        <li><a href="/page/${indexOfOldEntries}">π□ˆ¨ Older
posts</a></li>
                    <ul>
                </div>
            </#if>
            <footer>
                <h2>Copyright &copy; 2016 Alexandros Efthymiadis - Ioannis
Chaldaios B· All rights reserved</h2>
            </footer>
    </body>
    </html>
```

contact.ftl

```
    <!DOCTYPE html>
    <html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>My Blog</title>
```

```
        <!--http://osdir.com/ml/web.freemarker.user/2005-08/msg00004.html-->
        <style type="text/css">
        <#include "styles.css">
    </style>
</head>

<body>
        <div id=header>
            <h1>My Blog</h1>
        </div>
        <nav>
            <ul id="top-nav">
                <li><a href="/">HOME</a></li>
                <li><a href="/about">ABOUT</a></li>
                <li><a href="/contact">CONTACT</a></li>
                <#if username??>
                    <ul style="float:right;list-style-type:none;">
                        <li><a href="/newpost">NEW POST</a></li>
                        <li><a href="/logout">LOGOUT</a></li>
                    </ul>
                <#else>
                    <ul style="float:right;list-style-type:none;">
                        <li><a href="/login">LOGIN</a></li>
                        <li><a href="/signup">SIGN UP</a></li>
                    </ul>
                </#if>
            </ul>
        </nav>
        <div id="newpost">
            <form action="/contact" method="POST">
                <br>
                <#if username??>
                    <h2>Subject</h2>
                    <input class="new" type="text" name="subject" size="120"
value="${subject!""}">

                    <h2>Query</h2>
```

```
                    <textarea name="body" cols="120" rows="20">${body!""}</textarea>
                <#else>
                    <h2>Name</h2>
                    <input class="new" type="text" name="subject" size="120"
value="${subject!""}">

                    <h2>Email</h2>
                    <input class="new" type="text" name="tags" size="120"
value="${tags!""}">

                    <h2>Subject</h2>
                    <input class="new" type="text" name="subject" size="120"
value="${subject!""}">

                    <h2>Query</h2>
                    <textarea name="body" cols="120" rows="20">${body!""}</textarea>
                </#if>
                <br>
                <br>

                <input type="submit" value="SUBMIT">
                <p class="error"> ${errors!""}</p>
                <br>
                <br>
            </form>
        </div>
        <footer>
            <h2>Copyright &copy; 2016 Alexandros Efthymiadis - Ioannis
Chaldaios B· All rights reserved</h2>
        </footer>
    </body>
    </html>
```

entry_template.ftl

```
<!DOCTYPE html>
<html>
<head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>${post["title"]}</title>
          <!--http://osdir.com/ml/web.freemarker.user/2005-08/msg00004.html-->
        <style type="text/css">
        <#include "styles.css">
    </style>
        <script type="text/javascript">var switchTo5x=true;</script>
        <script type="text/javascript" src="http://w.sharethis.com/button/buttons.js"></script>
        <script type="text/javascript">stLight.options({publisher: "f9508b6a-a83d-4738-bb15-
37e27d5120f0", doNotHash: false, doNotCopy: false, hashAddressBar: false});</script>

</head>

<body>
        <div id=header>
                <h1>My Blog</h1>
        </div>
        <nav>
                <ul id="top-nav">
                        <li><a href="/">HOME</a></li>
                        <li><a href="/about">ABOUT</a></li>
                        <li><a href="/contact">CONTACT</a></li>
                        <#if username??>
                                <ul style="float:right;list-style-type:none;">
                                        <li><a href="/newpost">NEW POST</a></li>
                                        <li><a href="/logout">LOGOUT</a></li>
                                </ul>
                        <#else>
                                <ul style="float:right;list-style-type:none;">
                                        <li><a href="/login">LOGIN</a></li>
                                        <li><a href="/signup">SIGN UP</a></li>
                                </ul>
                        </#if>
                </ul>
```

```
        </nav>
        <div id="container">
            <div id="sidebar">
                <h2>SHARE</h2>
                <div class="navlist">
                    <ul>
                        <li><span class='st_facebook_large'
displayText='Facebook'></span></li>
                        <li><span class='st_twitter_large' displayText='Tweet'></span></li>
                        <li><span class='st_googleplus_large' displayText='Google
+'></span></li>
                        <li><span class='st_linkedin_large'
displayText='LinkedIn'></span></li>
                        <li><span class='st_email_large' displayText='Email'></span></li>
                    </ul>
                </div>
            </div>
            <div id="title">
                <h2>${post["title"]}</h2>
            </div>
            <div id="info">
                <b>Posted</b> ${post["date"]?datetime} | <b>By</b>
${post["author"]}<br>
            </div>
            <br>
            <div id="content">
                ${post["body"]}
            </div>
            <div id=tags>
                Tags:
                <#if post["tags"]??>
                    <#list post["tags"] as tag>
                        <a href="/tag/${tag}">${tag}</a>
                    </#list>
                </#if>
            </div>
            <div id="comments">
```

```
                <h2>Comments:</h2>
                <ul>
                    <#if post["comments"]??>
                        <#assign numComments = post["comments"]?size>
                    <#else>
                        <#assign numComments = 0>
                    </#if>
                    <#if (numComments > 0)>
                        <#list 0 .. (numComments -1) as i>
                            <form action="/like" method="POST">
                                <p>
                                    <input type="hidden" name="permalink",
value="${post["permalink"]}">
                                    <input type="hidden" name="comment_ordinal",
value="${i}">

                                    <span  class="author"><b>Author:
</b>${post["comments"][i]["author"]}</span>
                                    <br><br>
                                    <span class="likes">Likes:
${post["comments"][i]["num_likes"]}</span>
                                        
                                    <input type="submit" value="LIKE">
                                </p>
                            </form>
                            <br>
                            ${post["comments"][i]["body"]}
                            <br>
                        </#list>
                    </#if>
                </ul>
            </div>
            <div id="add-comment">
                    <br><hr><br><br>
                    <h2>Add a comment</h2>
                    <form action="/newcomment" method="POST">
```

```
                        <input type="hidden" name="permalink",
value="${post["permalink"]}">

                        <#if username??>
                            <textarea name="commentBody" cols="60"
rows="10">${comment["body"]}</textarea><br>
                        <#else>
                            <h3>Name (required)</h3>
                            <input type="text" name="commentName" size="60"
value="${comment["name"]}"><br>

                            <h3>Email (optional)</h3>
                            <input type="text" name="commentEmail" size="60"
value="${comment["email"]}"><br>

                            <h3>Comment</h3>
                            <textarea name="commentBody" cols="60"
rows="10">${comment["body"]}</textarea><br>
                        </#if>
                        <input type="submit" value="POST COMMENT">
                        <p class="error"> ${errors!""}</p>
                    </form>
            </div>
        </div>
        <div class="spacer"></div>
        <footer>
            <h2>Copyright &copy; 2016 Alexandros Efthymiadis - Ioannis
Chaldaios B· All rights reserved</h2>
        </footer>

    </body>
    </html>
```

error_template.ftl

```
<!DOCTYPE html>

<html>

<head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>Internal Error</title>
        <style type="text/css">
       <#include "styles.css">
    </style>
</head>

<body>
        <div id=header>
                <h1>My Blog</h1>
        </div>
        <nav>
                <ul id="top-nav">
                        <li><a href="/">HOME</a></li>
                        <li><a href="/about">ABOUT</a></li>
                        <li><a href="/contact">CONTACT</a></li>
                        <#if username??>
                                <ul style="float:right;list-style-type:none;">
                                        <li><a href="/newpost">NEW POST</a></li>
                                        <li><a href="/logout">LOGOUT</a></li>
                                </ul>
                        <#else>
                                <ul style="float:right;list-style-type:none;">
                                        <li><a href="/login">LOGIN</a></li>
                                        <li><a href="/signup">SIGN UP</a></li>
                                </ul>
                        </#if>
                </ul>
        </nav>

        <div id='error-text'>
                <span>500</span>
```

```
            <p>${error}</p>
            <p><input onclick="location.href = '/contact';"  type="submit" value="REPORT
PROBLEM"></p>
            <div class="spacer"></div>
            <footer>
                <h2>Copyright &copy; 2016 Alexandros Efthymiadis -
Ioannis Chaldaios B· All rights reserved</h2>
            </footer>
        </div>
    </body>
    </html>
```

login.ftl

```
<!DOCTYPE html>

<html>

<head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>Login</title>
        <style type="text/css">
       <#include "styles.css">
     </style>
</head>

<body>
        <div id=header>
            <h1>My Blog</h1>
        </div>
        <nav>
            <ul id="top-nav">
                <li><a href="/">HOME</a></li>
                <li><a href="/about">ABOUT</a></li>
                <li><a href="/contact">CONTACT</a></li>
```

```
                    <ul style="float:right;list-style-type:none;">
                        <li><a href="/login">LOGIN</a></li>
                        <li><a href="/signup">SIGN UP</a></li>
                    </ul>
                </ul>
            </nav>
            <div id="login">
                <form method="post">
                    <table>
                        <tr>
                            <td>
                                <input type="text" name="username" placeholder="Username"
value="${username}">
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <input type="password"  placeholder="Password"
name="password" value="">
                            </td>
                        </tr>
                    </table>
                    <input type="submit" value="SIGN IN">
                    <p class="error">${login_error}</p>
                </form>
                <p>Not a member? <a href="/signup">Sign up now</a></p>
            </div>
            <div class="spacer"></div>
            <div class="spacer"></div>
            <div class="spacer"></div>
            <footer>
                <h2>Copyright &copy; 2016 Alexandros Efthymiadis - Ioannis
Chaldaios B· All rights reserved</h2>
            </footer>
    </body>

    </html>
```

newpost_template.ftl

```
<!DOCTYPE html>

<html>

<head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>New post</title>
        <style type="text/css">
       <#include "styles.css">
    </style>
</head>

<body>
        <div id=header>
                <h1>My Blog</h1>
        </div>
        <nav>
                <ul id="top-nav">
                        <li><a href="/">HOME</a></li>
                        <li><a href="/about">ABOUT</a></li>
                        <li><a href="/contact">CONTACT</a></li>
                        <ul style="float:right;list-style-type:none;">
                                <li><a href="/newpost">NEW POST</a></li>
                                <li><a href="/logout">LOGOUT</a></li>
                        </ul>
                </ul>
        </nav>
        <div id="newpost">
                <form action="/newpost" method="POST">
                        <h2>Post Title</h2>
                        <input class="new" type="text" name="subject" size="120"
value="${subject!""}">
```

```
                <h2>Post Text</h2>
                <textarea name="body" cols="120" rows="20">${body!""}</textarea>

                <h2>Tags</h2>
                <input class="new" type="text" name="tags" placeholder="Comma separated,
please" size="120" value="${tags!""}"><br>
                <br>

                <input type="submit" value="CREATE POST">
                <p class="error"> ${errors!""}</p>
            </form>
        </div>
        <div class="spacer"></div>
        <footer>
            <h2>Copyright &copy; 2016 Alexandros Efthymiadis - Ioannis
Chaldaios B· All rights reserved</h2>
        </footer>
    </body>
    </html>
```

page_not_found.ftl

```
<!DOCTYPE html>

<html>

<head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>Internal Error</title>
        <style type="text/css">
       <#include "styles.css">
     </style>
</head>
```

```
<body>
    <div id=header>
        <h1>My Blog</h1>
    </div>
    <nav>
        <ul id="top-nav">
            <li><a href="/">HOME</a></li>
            <li><a href="/about">ABOUT</a></li>
            <li><a href="/contact">CONTACT</a></li>
            <#if username??>
                <ul style="float:right;list-style-type:none;">
                    <li><a href="/newpost">NEW POST</a></li>
                    <li><a href="/logout">LOGOUT</a></li>
                </ul>
            <#else>
                <ul style="float:right;list-style-type:none;">
                    <li><a href="/login">LOGIN</a></li>
                    <li><a href="/signup">SIGN UP</a></li>
                </ul>
            </#if>
        </ul>
    </nav>

    <div id='error-text'>
        <span>404</span>
        <p>${error}</p>
        <p><input onclick="location.href = '/';"  type="submit" value="Go back to the Home
page"></p>
        <div class="spacer"></div>
        <footer>
            <h2>Copyright &copy; 2016 Alexandros Efthymiadis -
Ioannis Chaldaios B· All rights reserved</h2>
        </footer>
    </div>
</body>
</html>
```

post_not_found.ftl

```
<!DOCTYPE html>

<html>

<head>
	<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
	<title>Page not found</title>
	<style type="text/css">
	<#include "styles.css">
   </style>
</head>

<body>
	<div id=header>
		<h1>My Blog</h1>
	</div>
	<nav>
		<ul id="top-nav">
			<li><a href="/">HOME</a></li>
			<li><a href="/about">ABOUT</a></li>
			<li><a href="/contact">CONTACT</a></li>
			<#if username??>
				<ul style="float:right;list-style-type:none;">
					<li><a href="/newpost">NEW POST</a></li>
					<li><a href="/logout">LOGOUT</a></li>
				</ul>
			<#else>
				<ul style="float:right;list-style-type:none;">
					<li><a href="/login">LOGIN</a></li>
					<li><a href="/signup">SIGN UP</a></li>
				</ul>
			</#if>
		</ul>
```

```
        </nav>

        <div id='error-text'>
            <span>404</span>
            <p>PAGE NOT FOUND</p>
            <p><input type="submit" value="REPORT PROBLEM"></p>
            <div class="spacer"></div>
            <footer>
                <h2>Copyright &copy; 2016 Alexandros Efthymiadis -
Ioannis Chaldaios B· All rights reserved</h2>
            </footer>
        </div>
    </body>
    </html>
```

posts_template.ftl

```
    <!DOCTYPE html>
    <html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>My Blog</title>
          <!--http://osdir.com/ml/web.freemarker.user/2005-08/msg00004.html-->
        <style type="text/css">
        <#include "styles.css">
      </style>
        <script type="text/javascript">var switchTo5x=true;</script>
        <script type="text/javascript" src="http://w.sharethis.com/button/buttons.js"></script>
        <script type="text/javascript">stLight.options({publisher: "f9508b6a-a83d-4738-bb15-
37e27d5120f0", doNotHash: false, doNotCopy: false, hashAddressBar: false});</script>

    </head>

    <body>
        <div id=header>
```

```
            <h1>My Blog</h1>
        </div>
        <nav>
            <ul id="top-nav">
                <li><a href="/">HOME</a></li>
                <li><a href="/about">ABOUT</a></li>
                <li><a href="/contact">CONTACT</a></li>
                <#if username??>
                    <ul style="float:right;list-style-type:none;">
                        <li><a href="/newpost">NEW POST</a></li>
                        <li><a href="/logout">LOGOUT</a></li>
                    </ul>
                <#else>
                    <ul style="float:right;list-style-type:none;">
                        <li><a href="/login">LOGIN</a></li>
                        <li><a href="/signup">SIGN UP</a></li>
                    </ul>
                </#if>
            </ul>
        </nav>
        <div id="container">
            <div id="sidebar">
                <h2>SHARE</h2>
                <div class="navlist">
                    <ul>
                        <li><span class='st_facebook_large'
displayText='Facebook'></span></li>
                        <li><span class='st_twitter_large' displayText='Tweet'></span></li>
                        <li><span class='st_googleplus_large' displayText='Google
+'></span></li>
                        <li><span class='st_linkedin_large'
displayText='LinkedIn'></span></li>
                        <li><span class='st_email_large' displayText='Email'></span></li>
                    </ul>
                </div>
            </div>
```

```
            <#list myposts as post>
                <div id="title">
                    <h2><a
href="/post/${post["permalink"]}">${post["title"]}</a></h2>
                </div>
                <br>
                <div id="content">
                    ${post["body"]!""}
                </div>
                <div id=tags>
                    Tags:
                    <#if post["tags"]??>
                        <#list post["tags"] as tag>
                            <a href="/tag/${tag}">${tag}</a>
                        </#list>
                    </#if>
                    |
                    Comments:
                    <#if post["comments"]??>
                        <#assign numComments = post["comments"]?size>
                            <#else>
                                <#assign numComments = 0>
                    </#if>
                    <a href="/post/${post["permalink"]}">${numComments}</a>
                </div>
            </#list>
        </div>
        <#if indexOfOldEntries??>
            <div class="pagination">
                <ul>
                    <li><a href="/page/${indexOfOldEntries}">π□ˆ¨ Older
posts</a></li>
                    <li><a href="/page/${indexOfNewEntries}">Newer posts
π ˆ□</a></li>
                <ul>
```

```
            </div>
        <#else>
            <div class="pagination"></div>
        </#if>
        <footer>
            <h2>Copyright &copy; 2016 Alexandros Efthymiadis - Ioannis Chaldaios B· All rights reserved</h2>
        </footer>
    </body>
    </html>
```

signup.ftl

```
<!DOCTYPE html>

<html>

<head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>Sign up</title>
        <style type="text/css">
       <#include "styles.css">
     </style>
</head>

<body>
        <div id=header>
            <h1>My Blog</h1>
        </div>
        <nav>
            <ul id="top-nav">
                <li><a href="/">HOME</a></li>
                <li><a href="/about">ABOUT</a></li>
                <li><a href="/contact">CONTACT</a></li>
                <ul style="float:right;list-style-type:none;">
```

```
                        <li><a href="/login">LOGIN</a></li>
                        <li><a href="/signup">SIGN UP</a></li>
                    </ul>
                </ul>
            </nav>
            <div id="login">
                <form method="post">
                    <table>
                        <tr>
                            <td>
                                <input type="text" name="firstname" placeholder="First
Name*" value="${firstName}">
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <input type="text" name="lastname" placeholder="Last Name*"
value="${lastName}">
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <input type="text" name="username" placeholder="Username*"
value="${username}">
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <input type="password"  placeholder="Password*"
name="password" value="">
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <input type="password"  placeholder="Verify Password*"
name="verify" value="">
                            </td>
```

```
                    </tr>
                    <tr>
                        <td>
                            <input type="text"  placeholder="Email*" name="email"
value="${email}">
                        </td>
                    </tr>
                </table>
                <input type="submit" value="GET STARTED">
                <p class="error">
                    ${firstName_error!""}
                    ${lastName_error!""}
                    ${username_error!""}
                    ${password_error!""}
                    ${verify_error!""}
                    ${email_error!""}
                </p>
            </form>
            <p>Already a member? <a href="/login">Login</a></p>
        </div>
        <footer>
            <h2>Copyright &copy; 2016 Alexandros Efthymiadis - Ioannis
Chaldaios B· All rights reserved</h2>
        </footer>
    </body>

    </html>
```

welcome.ftl

```
    <!DOCTYPE html>

    <html>

    <head>
```

```
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>Welocome back</title>
        <style type="text/css">
       <#include "styles.css">
     </style>
  </head>

  <body>
        <div id=header>
             <h1>My Blog</h1>
        </div>
        <nav>
             <ul id="top-nav">
                  <li><a href="/">HOME</a></li>
                  <li><a href="/about">ABOUT</a></li>
                  <li><a href="/contact">CONTACT</a></li>
                  <ul style="float:right;list-style-type:none;">
                       <li><a href="/newpost">NEW POST</a></li>
                       <li><a href="/logout">LOGOUT</a></li>
                  </ul>
             </ul>
        </nav>
        <div id="welcome">
             <p>Welcome Back, ${firstName}!</p>
        </div>
        <div class="spacer"></div>       <div class="spacer"></div>       <div
class="spacer"></div>
        <footer>
             <h2>Copyright &copy; 2016 Alexandros Efthymiadis - Ioannis
Chaldaios B· All rights reserved</h2>
        </footer>
  </body>
  </html>
```

6.1.3 CSS

styles.css

```
html, body {
	height: 100%;
	/* Required */
}

body {
	background: #EDEFF1;
	margin: 0;
	padding: 0;
	font: 90% "Trebuchet MS", verdana, arial, tahoma, sans-serif;
	line-height: 1.8em;
	color: #666;
}

#header {
	background-color: #388E8E;
	color: #EEEEEE;
	display: block;
	font-weight: normal;
	margin: 0;
	padding: 25px;
	text-align: center;
}

#top-nav {
	list-style-type: none;
	margin: 0;
	padding: 0;
	overflow: hidden;
	background-color: #2F4F4F;
	text-align: center;
}
```

```
#top-nav li {
    float: left;
}

#top-nav li a {
    display: block;
    color: white;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
}

#top-nav li a:hover {
    background-color: #388E8E;
}

#container {
    position: relative;
    min-height: 100%;
    width: 850px;
    margin: 50px auto;
    padding: 0 0 0 55px;
}

* html #container {
    height: 100%;
}

#sidebar {
    position: absolute;
    top: 0;
    bottom: 0;
    left: -10%;
    text-align: center;
    width: 200px;
    margin: 0 50px 0 0;
}
```

```
.navlist ul li {
	margin: 0 0 0 -42px;
	padding: 0;
	list-style-type: none;
	color: #BC5510;
	background: inherit;
}

#title {
	margin: 20px 0 -10px 130px;
	font: 130% verdana, "Trebuchet MS", arial, tahoma, sans-serif;
	padding: 5px;
	text-decoration: none;
	text-transform: uppercase;
	letter-spacing: 5px;
	color: #136CB2;
	background: inherit;
}

#info {
	padding: 7px;
	margin: 0 0 5px 130px;
	font: 100% verdana, "Trebuchet MS", arial, tahoma, sans-serif;
	color: #626262;
	background-color: #E0E0E0;
	border: medium none;
	vertical-align: baseline;
}

#content {
	padding: 10px;
	margin: 0 0 5px 130px;
	font: 100% verdana, "Trebuchet MS", arial, tahoma, sans-serif;
	color: #666;
	background: inherit;
	text-align: left;
```

```
}

#tags {
	padding: 10px;
	margin: 0 0 15px 130px;
	font: 100% verdana, "Trebuchet MS", arial, tahoma, sans-serif;
	border: 1px dashed #8a795d;
	font-size: 80%;
	color: #5A5A5A;
	background: inherit;
	text-align: center;
}

#comments {
	padding: 10px;
	margin: 0 0 15px 130px;
	font: 100% verdana, "Trebuchet MS", arial, tahoma, sans-serif;
	text-align: left;
}

#comments h2 {
	margin: 40px 0 -10px 0;
}

a {
	font-weight: bold;
	text-decoration: none;
}

a:link {
	color: #136CB2;
	text-decoration: none;
}

a:visited {
	color: #136CB2;
}
```

```
a:hover {
	color: #136CB2;
}

footer {
	margin: 40px 0 0 0;
	background-color: #388E8E;
	bottom: 0;
	box-shadow: 0 -1px 2px rgba(0, 0, 0, 0.4);
	height: 45px;
	left: 0;
	width: 100%;
	z-index: 100000;
	line-height: normal;
	text-align: center;
}

footer h2 {
	color: #EEEEEE;
	font-size: 12px;
	font-weight: normal;
	padding: 13px 0 0;
}

.spacer {
	padding: 30px;
}

.pagination {
	width: 500px;
	margin: auto;
	padding: 10px;
	font-size: 16px;
	text-align: center;
}
```

```
.pagination li {
	padding: 5px;
	list-style-type: none;
}

.pagination li a {
	color: #2F4F4F;
	text-align: center;
	padding: 14px 16px;
	text-decoration: none;
}

.pagination li a:hover {
	color: #388E8E;
	text-decoration: underline;
}


/* LOGIN - SIGN UP */

body div#login {
	margin: 0;
	padding: 25px;
	text-align: center;
}

form {
	margin-top: 30px;
	display: inline-block;
	text-align: center;
}

input[type="text"],
input[type="password"] {
	display: block;
	width: 309px;
	height: 35px;
```

```
        margin: 5px auto;
        background: #fff;
        border: 0px;
        padding: 5px;
        font-size: 16px;
        border: 2px solid #fff;
        transition: all 0.3s ease;
        border-radius: 5px;
        -moz-border-radius: 5px;
        -webkit-border-radius: 5px;
}

input[type="text"]:focus,
input[type="password"]:focus,
textarea:focus {
        border: 2px solid #1abc9d
}

input[type="submit"] {
        display: block;
        background: #1abc9d;
        width: 314px;
        padding: 12px;
        cursor: pointer;
        color: #fff;
        border: 0px;
        margin: auto;
        border-radius: 5px;
        -moz-border-radius: 5px;
        -webkit-border-radius: 5px;
        font-size: 17px;
        transition: all 0.3s ease;
}

input[type="submit"]:hover {
        background: #09cca6
}
```

```
.error {
    color: red
}

/* POST NOT FOUND */

body div#error-text {
    top: 30%;
    position: relative;
    font-size: 20px;
    text-align: center;
}

#error-text span {
    position: relative;
    background: #0A7189;
    color: #fff;
    font-size: 1200%;
    padding: 0 20px;
    border-radius: 5px;
    font-family: 'Roboto', sans-serif;
    , sans-serif;
    font-weight: bolder;
}

#error-text p {
    color: #2F4F4F;
    margin: 80px 0 -50px 0;
}

/* NEWPOST */

body div#newpost {
    top: 30%;
    font-size: 20px;
    text-align: center;
```

```
}

body div#newpost input[type="text"] {
    display: block;
    width: 800px !important;
    height: 35px;
    margin: 5px auto;
    background: #fff;
    border: 0px;
    padding: 5px;
    font-size: 16px;
    border: 2px solid #fff;
    transition: all 0.3s ease;
    border-radius: 5px;
    -moz-border-radius: 5px;
    -webkit-border-radius: 5px;
}

body div#newpost textarea {
    display: block;
    width: 800px !important;
    height: 300px;
    margin: 5px auto;
    background: #fff;
    border: 0px;
    padding: 5px;
    font-size: 16px;
    border: 2px solid #fff;
    transition: all 0.3s ease;
    border-radius: 5px;
    -moz-border-radius: 5px;
    -webkit-border-radius: 5px;
}

body div#newpost input[type="text"]:focus,
body div#add-comment input[type="text"]:focus,
body div#newpost input[type="password"]:focus,
```

```
body div#newpost textarea:focus,
body div#add-comment textarea:focus {
    border: 2px solid #1abc9d
}

body div#newpost h2 {
    text-align: left;
    width: 800px !important;
}

/* WELCOME */

body div#welcome {
    margin: 0;
    padding: 100px;
    font-size: 75px;
    text-align: center;
}

/* ABOUT */

body div#about {
    display: inline-block;
    margin: 0;
    margin-top: -30px;
    padding: 120px;
    margin-right: 100px;
    margin-left: 100px;
    font-size: 20px;
    text-align: center;
}

body div#about p {
    text-align: center;
}

/* ENTRY TEMPLATE */
```

```
body div#comments input[type="submit"] {
    display: inline;
    background: #1abc9d;
    width: 100px;
    padding: 8px;
    cursor: pointer;
    color: #fff;
    border: 0px;
    margin: auto;
    border-radius: 5px;
    -moz-border-radius: 5px;
    -webkit-border-radius: 5px;
    font-size: 10px;
    transition: all 0.3s ease;
}

.author {
    float: left;
    text-align: left;
}

.likes {
    text-align: left;
}

/* ADD COMMENT */

body div#add-comment {
    top: 30%;
    font-size: 15px;
    text-align: center;
}

body div#add-comment input[type="text"] {
    display: block;
    width: 500px !important;
```

```
        height: 35px;
        margin: 5px 0 -20px 0;
        background: #fff;
        border: 0px;
        padding: 5px;
        font-size: 16px;
        border: 2px solid #fff;
        transition: all 0.3s ease;
        border-radius: 5px;
        -moz-border-radius: 5px;
        -webkit-border-radius: 5px;
}

body div#add-comment textarea {
        display: block;
        width: 500px !important;
        height: 150px;
        margin: 5px auto;
        background: #fff;
        border: 0px;
        padding: 5px;
        font-size: 16px;
        border: 2px solid #fff;
        transition: all 0.3s ease;
        border-radius: 5px;
        -moz-border-radius: 5px;
        -webkit-border-radius: 5px;
}

body div#add-comment h2 {
        text-align: left;
        width: 500px !important;
        margin: 5px auto;
}

hr {
        border: 0;
```

```
        height: 1px;
        background-image: linear-gradient(to right, rgba(0, 0, 0, 0), rgba(0, 0,
0, 0.75), rgba(0, 0, 0, 0));
    }
```

6.2 Κώδικας Ελέγχου Απόδοσης

test.py

```
#!/usr/bin/python

import requests
import logging
import sys
import argparse
import datetime
import json
from fabric.api import local, settings, abort
from fabric.contrib.console import confirm

BLOG_SIGNUP_ENDPOINT = "http://{host}/signup"
BLOG_LOGIN_ENDPOINT = "http://{host}/login"
BLOG_NEWPOST_ENDPOINT = "http://{host}/newpost"
BLOG_NEWCOMMENT_ENDPOINT = "http://{host}/newcomment"
BLOG_GETPOST_ENDPOINT = "http://{host}/post"

HEADERS = {'Content-type': 'text/html; charset=utf-8'}

class TestBlog(object):
    def __init__(self):
        self.PARAMETERS = self.get_parameters()
        self.LOGGER = logging.getLogger('test')
        hdlr = logging.FileHandler(self.PARAMETERS['user'] + '.log')
        formatter = logging.Formatter('%(asctime)s - [%(levelname)s]
%(message)s')
        hdlr.setFormatter(formatter)
        self.LOGGER.addHandler(hdlr)
        self.LOGGER.setLevel(logging.INFO)

    def get_parameters(self):
        argparser = argparse.ArgumentParser('Run blog tests')
        argparser.add_argument('--host', '-H', help='Blog url.', required=False,
default='127.0.0.1:8082')
        argparser.add_argument('--data-set-size', '-D', help='Data set size.',
type=int, required=False, default=1)
        argparser.add_argument('--user', '-U', help='User prefix.',
required=False, default="test")
        argparser.add_argument('--read-only', '-R', help='Read only.',
action='store_true', default=False)
```

```
        argparser.add_argument('--write-only', '-W', help='Write only.',
action='store_true', default=False)
        argparser.add_argument('--read-write', '-A', help='Read-Write.',
action='store_true', default=False)
        argparser.add_argument('--comment', '-C', help='Add comments after every
new post.', action='store_true', default=False)
        argparser.add_argument('--prefix', nargs='+', help='Add comments after
every new post.', required=True)
        parameters = vars(argparser.parse_args())
        return parameters

    def user_signup(self, url, user):

        params = {
            'firstname': user,
            'lastname': user,
            'username': user,
            'email': user + '@' + user + '.' + user,
            'password': user,
            'verify': user,
        }

        r = requests.post(url, params = params, headers=HEADERS)
        if r.status_code == 200:
             self.LOGGER.info('Sign up user: ' + user)
        else:
            self.LOGGER.error('Fail to sign up user: ' + user + '. Status :' +
str(r.status_code))

    def user_login(self, url, user):

        params = {
            'username': user,
            'password': user,
        }
        r = requests.post(url, params = params, headers=HEADERS)
        if r.status_code == 200:
             self.LOGGER.info('Headers: ' + str(r.request.headers))
             self.LOGGER.info('Login User: ' + user + '. Session ID: ' +
str(r.request.headers['Cookie'][9:-1]))
             return str(r.request.headers['Cookie'][9:-1])
        else:
            self.LOGGER.error('User creation failed. Status code: ' +
r.status_code)
            return None

    def new_post(self, url, sessionID, sbj, body):
        cookie = {'session': str(sessionID)}

        params = {
            'subject': sbj,
            'body': body,
            'tags': 'tag',
        }
```

```
        r = requests.post(url, params = params, cookies=cookie,
allow_redirects=True)
        if r.status_code == 200:
            self.LOGGER.info('New post added with title: ' + sbj + '. Permalink for '
+ sbj + ':' + str(r.url))
            return r.url.split('/')[-1]
        else:
            self.LOGGER.error('Post creation failed. Session ID does not exist. Status
code: ' + str(r.status_code))
            return None

    def new_comment(self, url, sessionID, permalink):
        cookie = {'session': str(sessionID)}

        params = {
            'commentName': 'test2',
            'commentBody': 'This is a blog comment.',
            'permalink': permalink,
            'commentEmail': 'test2',
        }

        r = requests.post(url, params = params, cookies=cookie)

        if r.status_code == 200:
            self.LOGGER.info('Added comment in post: ' + permalink)
        else:
            self.LOGGER.error('Comment creation failed. Permalink does not exist. Status
code: ' + str(r.status_code))

    def get_post(self, url, sessionID, permalink):
        cookie = {'session': str(sessionID)}

        params = {
            'permalink': permalink,
        }

        r = requests.get(url, params = params, cookies=cookie)

        if r.status_code == 200:
            self.LOGGER.info('Added comment in post: ' + str(permalink) +
str(r.text()))
            self.LOGGER.info('Added comment in post: ' + str(permalink))
        else:
            self.LOGGER.error('Get post failed. Status code: ' +
str(r.status_code))

if __name__ == '__main__':
    D = TestBlog()

    starting_date_time = datetime.datetime.now()
    D.LOGGER.info(datetime.datetime.now())

    if D.PARAMETERS['write_only']:
        for i in range(1, D.PARAMETERS['data_set_size'] + 1):
            prefix = str(D.PARAMETERS['prefix'][i%4])
```

```
D.user_signup(BLOG_SIGNUP_ENDPOINT.format(**D.PARAMETERS), prefix +
D.PARAMETERS['user'])
            sessionID =
D.user_login(BLOG_LOGIN_ENDPOINT.format(**D.PARAMETERS), prefix +
D.PARAMETERS['user'])
            for j in range(1,5):
                sbj = prefix + D.PARAMETERS['user'] + str(i) + str(j)
                body = prefix + D.PARAMETERS['user'] + str(i) + str(j)
                permalink =
D.new_post(BLOG_NEWPOST_ENDPOINT.format(**D.PARAMETERS), sessionID,
sbj, body)
                if D.PARAMETERS['comment']:

D.new_comment(BLOG_NEWCOMMENT_ENDPOINT.format(**D.PARAMETERS),
sessionID, permalink)
    if D.PARAMETERS['read_only']:
        for i in range(1, D.PARAMETERS['data_set_size'] + 1):
            prefix = str(D.PARAMETERS['prefix'][i%4])

D.user_signup(BLOG_SIGNUP_ENDPOINT.format(**D.PARAMETERS), prefix +
D.PARAMETERS['user'])
            sessionID =
D.user_login(BLOG_LOGIN_ENDPOINT.format(**D.PARAMETERS), prefix +
D.PARAMETERS['user'])
            for j in range(1,5):
                sbj = prefix + D.PARAMETERS['user'] + str(i) + str(j)
                body = prefix + D.PARAMETERS['user'] + str(i) + str(j)

D.get_post(BLOG_GETPOST_ENDPOINT.format(**D.PARAMETERS) + '/' + sbj,
sessionID, sbj)
    if D.PARAMETERS['read_write']:
        for i in range(1, D.PARAMETERS['data_set_size'] + 1):
            prefix = str(D.PARAMETERS['prefix'][i%4])

D.user_signup(BLOG_SIGNUP_ENDPOINT.format(**D.PARAMETERS), prefix +
D.PARAMETERS['user'])
            sessionID =
D.user_login(BLOG_LOGIN_ENDPOINT.format(**D.PARAMETERS), prefix +
D.PARAMETERS['user'])
            for j in range(1,5):
                sbj = prefix + D.PARAMETERS['user'] + str(i) + str(j)
                body = prefix + D.PARAMETERS['user'] + str(i) + str(j)
                permalink =
D.new_post(BLOG_NEWPOST_ENDPOINT.format(**D.PARAMETERS), sessionID,
sbj, body)
                if D.PARAMETERS['comment']:

D.new_comment(BLOG_NEWCOMMENT_ENDPOINT.format(**D.PARAMETERS),
sessionID, permalink)

D.get_post(BLOG_GETPOST_ENDPOINT.format(**D.PARAMETERS) + '/' + sbj,
sessionID, permalink)

    D.LOGGER.info('Starting time: ' + str(starting_date_time))
```

```
    D.LOGGER.info('Duration for data set size ' + str(D.PARAMETERS['data_set_size']) +
':' + str((datetime.datetime.now() - starting_date_time)))
    #.strftime('%B %d, %Y'))
```

6.3 Ansible Deployment

hosts

```
[docker]
localhost

[mongodb]
localhost

[blog]
localhost
```

thesis.yml

```
---
- hosts: docker
  sudo: no
  roles:
    - { role: "docker-dnsmasq" }
  tags:
    - docker-dnsmasq

- hosts: blog
  sudo: no
  roles:
    - { role: "blog", action: "cleanup" }
  tags:
    - clean

- hosts: blog
  sudo: no
  gather_facts: yes
  roles:
    - { role: "blog", action: "build" }
  tags:
    - build
    - blog-cluster

- hosts: mongodb
  sudo: no
  roles:
    - { role: "mongodb", action: "deploy" }
  tags:
    - mongodb
```

```
- hosts: mongodb
  sudo: no
  roles:
    - { role: "mongodb", action: "cluster" }
  tags:
    - mongodb-cluster


- hosts: blog
  sudo: no
  roles:
    - { role: "blog", action: "deploy" }
  tags:
    - blog

- hosts: blog
  sudo: no
  gather_facts: yes
  roles:
    - { role: "blog", action: "deploy-cluster" }
    - { role: "blog", action: "haproxy" }
  tags:
    - deploy-cluster
    - blog-cluster
```

roles/blog/tasks/main.yml

```
---

- include: "{{ action|default(deploy) }}.yml"
```

roles/blog/tasks/cleanup.yml

```
- name: Debug containers
  shell: docker ps -a
  tags:
    - clean

- name: Cleanup docker containers
  shell: docker stop `docker ps -a -q` && docker rm `docker ps -a -q`
  ignore_errors: true
  tags:
    - clean
```

roles/blog/tasks/deploy-cluster.yml

```
---

- name: Deploy blog (1st container)
  shell: "docker run -ti -d -p \"8090:8082\" --add-host mongos:{{ ansible_eno1.ipv4.address }} -e
\"JAVA_HOME=/opt/jdk/jdk1.8.0_151\" --name blog1 java-app:{{ blog_version }}"
```

```
 tags:
  - deploy-cluster

- name: Deploy 2nd blog (2nd container)
 shell: "docker run -ti -d -p \"8091:8082\" --add-host mongos:{{ ansible_eno1.ipv4.address }} -e
\"JAVA_HOME=/opt/jdk/jdk1.8.0_151\" --name blog2 java-app:{{ blog_version }}"
 tags:
  - deploy-cluster
```

roles/blog/tasks/deploy.yml

```
---


- name: Deploy blog
 shell: "docker run -ti -d -p \"8082:8082\" --name blog java-app:{blog_version}"
 tags:
  - deploy
```

roles/blog/tasks/haproxy.yml

```
---

- name: Deploy HAProxy
 shell: "docker run -ti -d -p \"8082:8082\" --name haproxy myhaproxy"
 tags:
  - deploy-cluster
```

roles/blog/templates/Dockerfile.j2

```
FROM jetty:jre8

COPY apache-maven-3.3.3-bin.tar.gz /opt/apache-maven-3.3.3-bin.tar.gz

RUN cd /opt && tar -xvf apache-maven-3.3.3-bin.tar.gz && ln -s apache-
maven-3.3.3/bin/mvn mvn && chmod 0700 mvn && mkdir mongo-project

RUN apt-get -y update && apt-get -y install software-properties-
common
RUN apt-get -y install python-software-properties

RUN mkdir /opt/jdk && cd /opt && \
  wget --header "Cookie: oraclelicense=accept-securebackup-cookie"
http://download.oracle.com/otn-pub/java/jdk/8u151-
b12/e758a0de34e24606bca991d704f6dcbf/jdk-8u151-linux-x64.tar.gz && \
  tar -zxf jdk-8u151-linux-x64.tar.gz -C /opt/jdk && \
  update-alternatives --install /usr/bin/java java
/opt/jdk/jdk1.8.0_151/bin/java 100 && \
  update-alternatives --install /usr/bin/javac javac
/opt/jdk/jdk1.8.0_151/bin/javac 100
```

```
WORKDIR /opt/mongo-project

COPY mongo-project /opt/mongo-project

ENV JAVA_HOME="/opt/jdk/jdk1.8.0_151"
EXPOSE 8082

CMD /opt/mvn compile exec:java -Dexec.mainClass=blog.BlogController -
Dexec.args="mongodb://mongos:27000" && tail -f /opt/mongo-project/pom.xml
```

roles/blog/tasks/build.yml

```
---

- name: Temporary directory
  shell: mktemp -d
  register: mktemp
  tags:
    - build

- name: Create Dockerfile
  template: src=Dockerfile.j2 dest={{ mktemp.stdout }}/Dockerfile
  tags:
    - build

- name: Copy apache
  copy: src=/opt/apache-maven-3.3.3-bin.tar.gz dest={{ mktemp.stdout }}/

- name: Copy project
  copy: src=../mongo-project dest={{ mktemp.stdout }}
  tags:
    - build

- name: Build blog
  shell: "cd {{ mktemp.stdout }} && docker build -t java-app:{{ blog_version }} ."
  tags:
    - build

- name: Remove temporary directory
  file: path={{ mktemp.stdout }} state=absent
  tags:
    - build
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence 1st Edition

[2] Types of NoSQL Databases - https://www.mongodb.com/scale/types-of-nosql-databases

[3] Introducing NoSQL and MongoDB - http://www.informit.com/articles/article.aspx?p=2247310

[4] Tutorials Point - MongoDB - Advantages. URL: https://www.tutorialspoint.com/mongodb/mongodb_advantages.htm

[5] MongoDB Inc. “MongoDB Java Driver”. URL: http://mongodb.github.io/mongo-java-driver/3.2/

[6] MongoDB Inc. “The MongoDB 3.2 Manual”. URL: https://docs.mongodb.com/v3.2/

[7] Architecting for Scale, High Availability for Your Growing Applications By Lee Atchison

[8] Εισαγωγή στη Java - Λιακέας Γιώργος

[9] Programming APIs with the Spark Web Framework - Kristopher Sandoval, Travis Spencer, Per Wendel

[10] “FreeMarker Java Template Engine” - Apache Software Foundation (ASF). URL: https://freemarker.apache.org/

[11] “What is Maven” - Apache Software Foundation (ASF). URL: https://maven.apache.org/what-is-maven.html

[12] “Guide to naming conventions on groupId, artifactId and version” - Apache Software Foundation (ASF). URL: https://maven.apache.org/guides/mini/guide-naming-conventions.html

[13] “What is distributed computing” - IBM Knowledge Center. URL: https://www.ibm.com/support/knowledgecenter/en/SSAL2T_8.2.0/com.ibm.cics.tx.doc/concepts/c_wht_is_distd_comptg.html

[14] A brief introduction to distributed systems By Maarten van Steen · Andrew S. Tanenbaum

[15] Docker Documentation - Docker Inc. URL: https://docs.docker.com/

[16] The python Wiki - Python Software Foundation. URL:

https://wiki.python.org/moin/FrontPage

[17] Overview of Docker Compose – Docker Inc. URL: https://docs.docker.com/compose/overview/

[18] Overview How Aansible Works - Red Hat, Inc. . URL: https://www.ansible.com/overview/how-ansible-works

[19] An Introduction to HAProxy and Load Balancing Concepts - DigitalOcean Inc. URL: https://www.digitalocean.com/community/tutorials/an-introduction-to-haproxy-and-load-balancing-concepts