ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Πτυχιακή Εργασία

Κατανεμημένη Διαχείριση και Διασφάλιση Προσωπικών Δεδομένων

Φλωρίνα Μπιλέτσιου

ΑΜ: 2114033

Επιβλέπων καθηγητής: Κολομβάτσος Κωνσταντίνος

Λαμία, 2021

Περίληψη

Δεδομένου ότι τα κατανεμημένα συστήματα είναι σήμερα μια από τις πιο ευρέως χρησιμοποιούμενες τεχνολογίες σε όλες τις μεγάλες και καθημερινές εφαρμογές, και ταυτόχρονα ο όγκος διαθέσιμων πληροφοριών προς διαχείριση αυξάνεται συνεχώς, τα παραπάνω ζητήματα παραμένουν ένας ιδιαίτερα σημαντικός τομέας έρευνας και ανάπτυξης. Αυτό εξηγείται εν μέρει από τα πολλαπλά χαρακτηριστικά και την πολυπλοκότητα τέτοιων συστημάτων όπου λαμβάνονται υπ' όψιν κατά την ανάπτυξη σύγχρονων εφαρμογών και συστημάτων.

Ακόμη, παρατηρείται πως πλέον για την επεξεργασία δεδομένων επιλέγεται η κατανομή των πληροφοριών σε πολλά συστήματα υπολογιστών, η οποία τεχνική προσφέρει πολλά πλεονεκτήματα έναντι των εφαρμογών ενός μοναδικού κόμβου και παλαιότερες τεχνολογίες.

Ο στόχος αυτής της εργασίας είναι να δοθεί μια σαφή εικόνα των κατανεμημένων συστημάτων, συμπεριλαμβανομένου του ορισμού τους, οι τεχνικές σχεδιασμού τους και μερικές από τις πιο κοινές κατηγορίες κατανεμημένων συστημάτων. Επιπλέον, αξιολογείται το θέμα της κατανομής δεδομένων με συνοπτικό τρόπο, συμπεριλαμβανομένης της περιγραφής του, των μεθόδων κατανομής δεδομένων, πλεονεκτημάτων και αρχών σχεδιασμού. Ενώ παράλληλα επισημαίνεται η σημασία της κατανεμημένης επεξεργασίας και αποθήκευσης δεδομένων αλλά και ο ρόλος της ιδιοκτησίας των πληροφοριών μεταξύ χρηστών.

Τέλος, ακολουθεί το εγχειρίδιο μιας εφαρμογής κατανεμημένης διαχείρισης και διασφάλισης προσωπικών δεδομένων, η οποία αναπτύχθηκε για τους σκοπούς αυτής της μελέτης χρησιμοποιείται για την επίδειξη όλων των παραπάνω θεμάτων και μελλοντικές προεκτάσεις αυτής.

Λέξεις – Κλειδιά

Κατανεμημένα Συστήματα, Κατανεμημένη Διαχείριση Δεδομένων, Κατανεμημένες Βάσεις Δεδομένων

Abstract

As distributed systems are today one of the most widely used technologies in all large and everyday used applications, and at the same time the volume of information available for management is constantly increasing, the above issues remain a particularly important area of research and development. This is partly explained by the multiple characteristics and complexity of such systems where they are considered when developing modern applications and systems.

Furthermore, it is noted that for the task of data processing the distribution of information is chosen in many computer systems, which technique offers many advantages over the applications of a single node and older technologies.

The aim of this work is to give a clear picture of distributed systems, including their definition, their design techniques and some of the most common categories of distributed systems. In addition, the issue of data distribution is assessed, including its description, data distribution methods, advantages, and design principles. While at the same time the importance of distributed data processing and storage is highlighted, as well as the role of information ownership among users.

Finally, the manual of a distributed data management and security application is following, which was developed for the purposes of this study, and it is used to demonstrate all the above issues and its possible future implications.

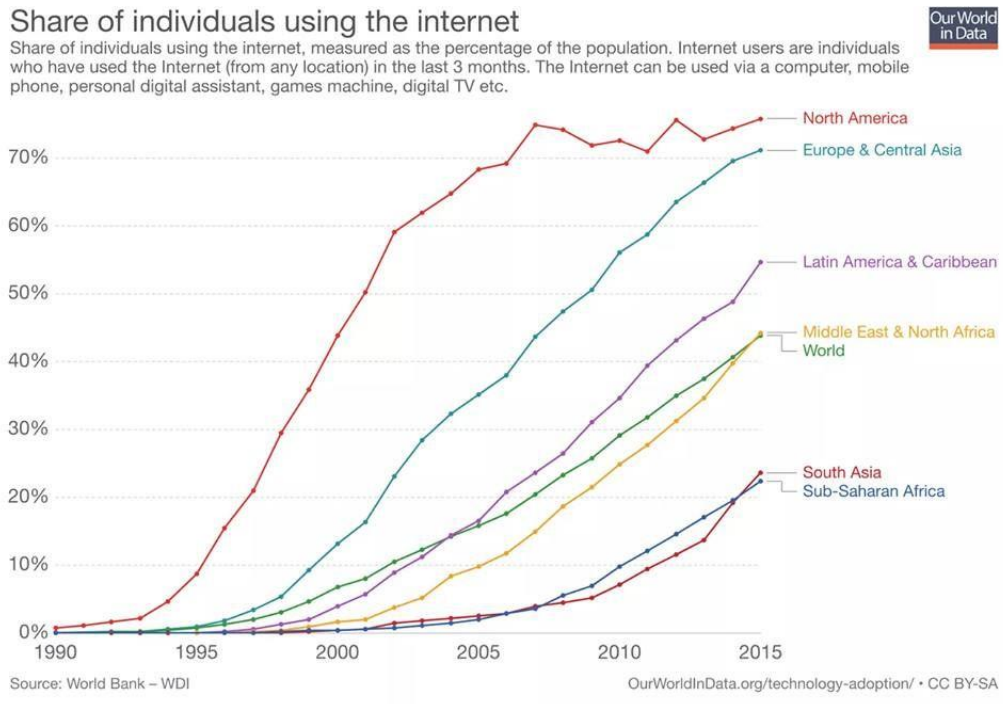
Keywords

Distributed Systems, Distributed Database Management, Distributed Databases

Περιεχόμενα

1. Εισαγωγή

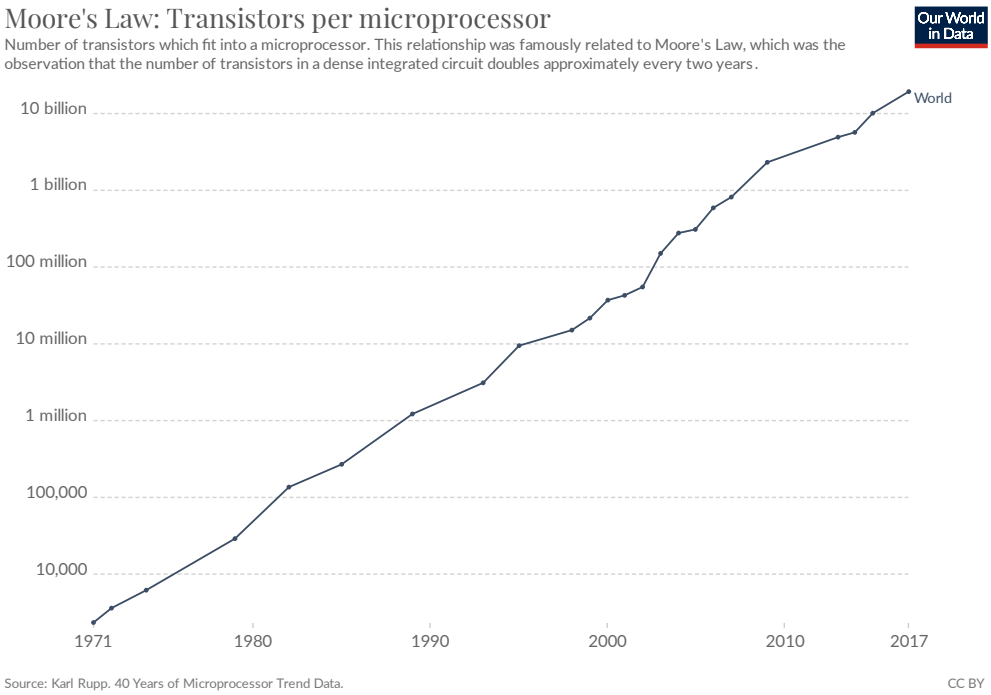
Τα τελευταία χρόνια είναι γεγονός πως η ανάγκη για πιο αποδοτικά συστήματα τα οποία προσφέρουν τη δυνατότητα κατανεμημένης αποθήκευσης δεδομένων και διαχείρισης αυτών είναι επιτακτική καθώς ολοένα και παραπάνω τομείς της καθημερινότητας συσχετίζονται με τη χρήση του διαδικτύου και την καταγραφή των προσωπικών ή ακόμα και ευαίσθητων δεδομένων των χρηστών. Σύμφωνα με μια σχετική μελέτη που πραγματοποιήθηκε από τον ερευνητή Charles Babcock, κάθε 12 έως 18 μήνες, ο όγκος δεδομένων της μέσης εταιρείας σχεδόν διπλασιάζεται (Babcock, 2006). Αναλογιζόμενοι τη συνεχή αύξηση της χρήσης των μηχανών αναζήτησης, των τηλεδιασκέψεων, τις συναλλαγές μετοχών που πραγματοποιούνται πλέον κατά κόρον διαδικτυακά, την ηλεκτρονικοποίηση των τραπεζικών υπηρεσιών, τη χρήση ηλεκτρονικών κοινωνικών δικτύων ως βασικό μέσο κοινωνικών επαφών, τις ιατρικές υπηρεσίες που προσφέρονται ηλεκτρονικά γίνεται φανερό πως οι υπηρεσίες που βασίζονται στο διαδίκτυο διαδραματίζουν κυρίαρχο ρόλο όπως είναι εμφανές στην Εικόνα 1 παρακάτω.



Εικόνα 1: Individuals using the Internet (% of population), image downloaded from https://ourworldindata.org/technology-adoption at May 2021.

Πρόσφατα σχηματίστηκαν και καθιερώθηκαν ανάλογοι νόμοι και κανονισμοί όσον αφορά τη διαχείριση των δεδομένων των χρηστών, γεγονός που επισήμανε τη βαρύτητα του ζητήματος. Πλέον η κάθε επιχείρηση ή οργανισμός ωφελεί να τηρεί τους κανόνες του Γενικού Κανονισμού Προστασίας Δεδομένων (GDPR) και επιβεβαιώνει πως ο χρήστης του συστήματος κατέχει την ιδιοκτησία των δεδομένων του, επιλέγει ο ίδιος τον τρόπο με τον οποίο τα δεδομένα του χρησιμοποιούνται ή έστω έχει επίγνωση της διαχείρισης των πληροφοριών του από τον κάθε οργανισμό (Ghosh, 2014).

Ο ρυθμός ανάπτυξης των υπολογιστικών συστημάτων ήταν και παραμένει ραγδαίος. Από το 1945 όπου θεωρείται πως ξεκίνησε η σύγχρονη εποχή των υπολογιστών, μέχρι και το 1985, τα διαθέσιμα υπολογιστικά συστήματα ήταν ακριβά και μεγάλα σε διαστάσεις. Επίσης η συνδεσιμότητα, για την ακρίβεια η έλλειψη αυτής περιόρισε τα υπολογιστικά συστήματα και τα ανάγκασε να λειτουργούν αποκλειστικά ανεξάρτητα. Τα επόμενα χρόνια ωστόσο, αναπτύχθηκαν δύο βασικές τεχνολογικές καινοτομίες που άλλαξαν τον τρόπο χρήσης των υπολογιστών ριζικά. Η πρώτη πρόοδος που παρατηρήθηκε ήταν η ανάπτυξη ισχυρών μικροεπεξεργαστών (microprocessors), η οποία οδήγησε στη δημιουργία των επεξεργαστών πολλαπλών πυρήνων (multicore CPUs). Η δεύτερη ανακάλυψη ήταν τα δίκτυα υπολογιστών υψηλής ταχύτητας. Πιο συγκεκριμένα, τα δίκτυα τοπικής περιοχής (Local-area networks, LANs) επιτρέπουν τη σύνδεση χιλιάδων συσκευών που βρίσκονται στο ίδιο κτίριο σε ένα τοπικό δίκτυο το οποίο επιτρέπει τη μεταφορά μικρών ποσοτήτων πληροφοριών σε διάστημα μικροδευτερολέπτων. Όσο αφορά τα δίκτυα ευρείας περιοχής (Wide-area networks, WANs) παρέχουν τη δυνατότητα εκατοντάδες εκατομμύρια μηχανές να συνδέονται παγκοσμίως σε ταχύτητες που κυμαίνονται από δεκάδες χιλιάδες έως εκατοντάδες εκατομμύρια bps (bits per second). Παράλληλα με τη βελτιστοποίηση της δικτύωσης, έχει παρατηρηθεί η ελαχιστοποίηση του μεγέθους των συστημάτων υπολογιστών η οποία οδήγησε στη σημερινή έκδοση των κινητών τηλεφώνων (smartphones). Όπως γίνεται φανερό από την Εικόνα 2, σύμφωνα με το Νόμο του Moore, οι υπολογιστές γίνονται ολοένα και μικρότεροι σε μέγεθος, αλλά ταυτόχρονα πιο ισχυροί και αποδοτικοί.



Εικόνα 2: Moore's Law: Transistors per microprocessor, image download from https://ourworldindata.org/grapher/transistors-per-microprocessor at May 2021

Το ίδιο ισχύει και για τα smartphone, τα οποία σήμερα είναι γεμάτα με αισθητήρες, πολλή μνήμη και μια ισχυρή CPU, αυτές οι συσκευές δεν είναι τίποτα λιγότερο από πλήρεις υπολογιστές. Το αποτέλεσμα όλων αυτών των τεχνολογιών είναι ότι είναι εφικτό να δημιουργηθεί ένα υπολογιστικό σύστημα που αποτελείται από πολλούς δικτυωμένους υπολογιστές, οι οποίοι συνήθως διασκορπίζονται γεωγραφικά και το γεγονός αυτό καθιστά ολόκληρο το σύστημα ένα κατανεμημένο σύστημα. Ένα κατανεμημένο σύστημα μπορεί να ποικίλει σε μέγεθος από λίγες συσκευές έως εκατομμύρια συσκευές, στη μέθοδο διασύνδεσης από ασύρματο έως ενσύρματο ή ακόμα και τα δύο. Τέλος, τα κατανεμημένα συστήματα είναι πολύ δυναμικά και ρευστά όσον αφορά την τοπολογία τους. Ένας υπολογιστής μπορεί να συμμετάσχει ή να αποχωρήσει από το σύστημα ανά πάσα στιγμή, πράγμα που σημαίνει ότι η απόδοση και το υποκείμενο δίκτυο μπορεί να αλλάζει συνεχώς (Steen & Tanenbaum , 2016).

2. Κατανεμημένα Συστήματα και Εφαρμογές

Αρχικά, ως κατανεμημένο σύστημα ορίζεται: «Το κατανεμημένο σύστημα είναι μια συλλογή από αυτόνομους υπολογιστές που συνδέονται μεταξύ τους μέσω ενός δικτύου και χρησιμοποιούν ειδικά σχεδιασμένο λογισμικό για την παροχή ενοποιημένων υπολογιστικών υπηρεσιών.» (Coulouris, Dollimore, Kindberg, & Blair, 2001). Σήμερα, η πλειοψηφία των εμπορικών και μεγάλων συστημάτων είναι κατανεμημένα, ώστε να μπορούν να ανταπεξέλθουν στις σύγχρονες απαιτήσεις. Ενώ στα πρώτα στάδια η κατανομή του φόρτου γινόταν εσωτερικά σε μια υπολογιστική μηχανή, πλέον η επεξεργασία των πληροφοριών επιβάλλει τη χρήση πολλαπλών υπολογιστών.

2.1 Χαρακτηριστικά Κατανεμημένων Συστημάτων

Η τεχνική των κατανεμημένων συστημάτων δύναται να διαφέρει από εφαρμογή σε εφαρμογή, πάραυτα υπάρχουν κάποια βασικά χαρακτηριστικά που περιγράφουν τα κατανεμημένα συστήματα.

Πιο συγκεκριμένα:

- Διαμοιρασμός πόρων: Κοινή χρήση πόρων υλικού, λογισμικού και δεδομένων.
- Ανοικτή λειτουργία: Χρήση εξοπλισμού και λογισμικού από διάφορους κατασκευαστές.
- Παράλληλη λειτουργία – Ταυτοχρονισμός: Ταυτόχρονη επεξεργασία για τη βελτίωση της απόδοσης.
- Επεκτασιμότητα: Αυξημένη διεκπεραιωτή ικανότητα με την προσθήκη νέων πόρων.
- Ανοχή σφαλμάτων: η δυνατότητα συνέχειας της λειτουργίας αφού ανακύψει κάποιο σφάλμα.

2.1.1 Στόχοι των Κατανεμημένων Συστημάτων

Αρχικά, πριν σχεδιαστεί και αναπτυχθεί κάποιο σύστημα πρέπει να αναλογιστεί η χρησιμότητα της οποιαδήποτε κατανομής. Ανάλογα τη φύση και το σκοπό που θα επιτελεί η κάθε εφαρμογή, είναι πιθανό πως η κατανομή του συστήματος δεν είναι η ιδανική επιλογή. Στο παρόν τμήμα θα αναφερθώ στους τέσσερις βασικούς στόχους που πρέπει να τηρούνται ώστε να αξίζει η προσπάθεια που απαιτείται για την κατανόηση του συστήματος.

1. Προσβασιμότητα των Πόρων

   Ο βασικός στόχος ενός κατανεμημένου συστήματος είναι να διευκολύνει τους χρήστες να αποκτούν απομακρυσμένη πρόσβαση σε πόρους και στην ελεγχόμενη και αποδοτικότερη κατανομή αυτών. Ως διαθέσιμος πόρος μπορούν να θεωρηθούν πόροι πολλών φύσεων, όπως για παράδειγμα εκτυπωτές, υπολογιστικά συστήματα, αποθηκευτικές μονάδες, δεδομένα, αρχεία, δίκτυα, κτλ. Ο κύριος λόγος που η κατανομή τους είναι τόσο συνηθισμένη πρακτική είναι πως πρόκειται για οικονομικότερη επιλογή. Ο διαμοιρασμός πόρων όπως υπολογιστική δύναμη, αποθηκευτικές μονάδες υψηλής αποδοτικότητας, κ.α. μπορεί να παρέχει όλες αυτές τις υπηρεσίες οικονομικότερα. Επίσης, τα προβλήματα λόγω γεωγραφικών περιορισμών απαλείφονται.

2. Διαφάνεια κατανομής

   Κατά την ανάπτυξη ενός κατανεμημένου συστήματος ιδιαίτερα σημαντικό θεωρείται το χαρακτηριστικό της διαφάνειας. Με τον όρο «διαφάνεια» περιγράφεται η προσπάθεια να μην είναι φανερό πως υπάρχει ή οποιαδήποτε κατανομή κατά τη χρήση της κατανεμημένης εφαρμογής. Το οποιαδήποτε σύστημα επιτυγχάνει να παρουσιαστεί στους χρήστες ως ένα μοναδικό υπολογιστικό σύστημα θεωρείται πως επιτυγχάνει την επιθυμητή διαφάνεια.
   Αξίζει ακόμα να τονιστεί πως υπάρχουν διάφορα είδη διαφάνειας στα κατανεμημένα συστήματα, αναφορικά: διαφάνεια όσον αφορά τη γεωγραφική κατανομή των πόρων, απόκρυψη της αναπαράστασης των δεδομένων και γενικότερα τον τρόπο πρόσβασης στην πηγή των πληροφοριών, επίσης η πιθανή μετανάστευση ή κλωνοποίηση πόρων (είτε τη στιγμή που χρησιμοποιείται είτε όταν δεν είναι σε χρήση άμεσα) είναι μια διαδικασία που αποκρύπτεται, απόκρυψη της κοινής χρήσης μιας πηγής από πολλαπλούς χρήστες και τέλος είναι πολύ σημαντική η διατήρηση της διαφάνειας σχετικά με τις πιθανές αποτυχίες και την αναμενόμενη ανάκαμψη του συστήματος.
   Ακόμα, ενώ η διαφάνεια στα κατανεμημένα συστήματα έχει πολλά πλεονεκτήματα, ο ιδανικός βαθμός διαφάνειας εξαρτάται άμεσα από τη φύση και τις ανάγκες της κάθε εφαρμογής. Για μερικά συστήματα η πλήρη απόκρυψη των κατανεμημένων πτυχών είναι δυνατόν να δημιουργεί επιπλέον δυσκολίες.

Συνεπώς, η διαφάνεια κατανομής αποτελεί έναν κατά γενική ομολογία καλό στόχο κατά τη σχεδίαση και την εφαρμογή κατανεμημένων συστημάτων, αλλά θα πρέπει να εξετάζεται σε συνδυασμό με άλλα ζητήματα όπως η απόδοση και η σαφήνεια.

3. Διαφάνεια – Ανοιχτός κώδικας

   Ένας σημαντικός στόχων των κατανεμημένων συστημάτων είναι να πραγματοποιούνται οι υπηρεσίες που υπόσχεται πως προσφέρει. Απαιτείται αναλυτική εξήγηση αλλά και οδηγίες της διαμόρφωσης του.

   Επίσης, ο σχεδιασμός του συστήματος θα πρέπει να είναι τέτοιος, όπου η οποιαδήποτε αλλαγή σε κάποιο μέρος της εφαρμογής θα είναι δυνατή. Πιθανές αλλαγές αποτελούν η αντικατάσταση, επέκταση ή μετακίνηση κάποιου μέρους του συστήματος.

   Οι περισσότερες παλαιότερες εφαρμογές υλοποιήθηκαν με μια πιο μονολιθική προσέγγιση, η οποία καθιστά την οποιαδήποτε αλλαγή σημαντικά δύσκολη καθώς επηρεάζει άμεσα τη λειτουργία του συστήματος. Αυτά τα συστήματα χαρακτηρίζονται ως κλειστά.

4. Επεκτασιμότητα

   Χωρίς αμφιβολία, με τη ραγδαία ανάπτυξη της τεχνολογίας και την αυξανόμενη χρήση του διαδικτύου, η δυνατότητα επεκτασιμότητας για ένα σύστημα αποτελεί έναν από τους βασικούς στόχους κατά το σχεδιασμό.

   Σύμφωνα με το Neuman (1994), η επεκτασιμότητα ενός συστήματος μπορεί να μετρηθεί σε τουλάχιστον τρεις διαστάσεις (Neuman, 1994). Αρχικά, ένα σύστημα μπορεί να είναι επεκτάσιμο σε σχέση με το μέγεθος του, το οποίο σημαίνει πως είναι δυνατή η προσθήκη επιπλέον χρηστών ή πόρων στο σύστημα. Επιπλέον, όταν αναφερόμαστε σε ένα γεωγραφικά επεκτάσιμο σύστημα, γίνεται λόγος για ένα σύστημα στο οποίο οι χρήστες μπορεί να βρίσκονται γεωγραφικά ¨μακριά¨ από τους πόρους, χωρίς η προαναφερθείσα απόσταση να επηρεάζει την εμπειρία. Τέλος, ένα σύστημα μπορεί να είναι διοικητικά επεκτάσιμο, το οποίο σημαίνει πως η διαχείριση του είναι εύκολη ακόμα και αν εκτείνεται για πολλαπλούς διοικητικούς οργανισμούς.

Αξίζει να τονιστεί πως, κατά την προσπάθεια ένα σύστημα να είναι επεκτάσιμο εμφανίζονται κάποια εμπόδια τα οποία πρέπει να ληφθούν υπ' όψιν. Παραδείγματος χάριν, όταν ένα σύστημα επεκτείνεται σε σχέση με το μέγεθος του χρειάζεται να αντιμετωπιστούν οι περιορισμοί που σχηματίζονται λόγω των κεντρικοποιημένων υπηρεσιών, δεδομένων ή/και αλγορίθμων (Tanenbaum & Steen, 2006).

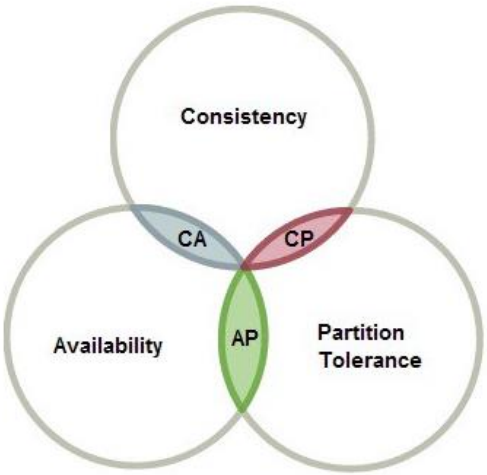
Η γεωγραφική επεκτασιμότητα επίσης αντιμετωπίζει κάποια εμπόδια, πιο συγκεκριμένα το συνηθέστερο εμπόδιο είναι πως η πλειονότητα των υπαρχόντων κατανεμημένων συστημάτων έχει σχεδιαστεί για τη χρήση τοπικών δικτύων βασισμένα σε σύγχρονη επικοινωνία. Με τη χρήση σύγχρονης επικοινωνίας, ένα μέρος του συστήματος, που γενικά αναφέρεται ως χρήστης, μπλοκάρει την επικοινωνία μέχρι να λάβει απάντηση. Σε ένα τοπικό δίκτυο αυτή η συμπεριφορά δεν προκαλεί κάποιο πρόβλημα λόγω των σχεδόν στιγμιαίων ταχυτήτων που ένα τοπικό δίκτυο προσφέρει, αλλά είναι εύκολα αντιληπτό πως σε ένα σύστημα ευρείας περιοχής θα προκληθούν σημαντικές καθυστερήσεις.

Τέλος, ένα δύσκολο πρόβλημα, το οποίο μπορεί να παραμείνει ως ανοιχτή ερώτηση σε πολλές περιπτώσεις, σχετίζεται με την κλιμάκωση ενός κατανεμημένου συστήματος σε πολλές ανεξάρτητες διοικητικές περιοχές. Πρέπει να λυθεί το ζήτημα της μη-συμβατότητας μεταξύ πολιτικών, τη χρήση πόρων, τις τεχνικές διαχείρισης αλλά και την ασφάλεια (Tanenbaum & Steen, 2006, σσ. 11-12).

2.1.2 Θεώρημα CAP

Είναι γεγονός πως ο σχεδιασμός ενός κατανεμημένου συστήματος αποτελεί μια περίπλοκη διαδικασία που απαιτεί ιδιαίτερη προσοχή έτσι ώστε να συσσωρευτούν οι γνώσεις και εμπειρίες των τελευταίων ετών, οι απαραίτητες γνώσεις και προβλέψεις για σύγχρονες τεχνολογίες αλλά και να ληφθεί υπ' όψιν το τελικό κόστος.

Το θεώρημα CAP (CAP theorem), το οποίο σχεδιάστηκε και παρουσιάστηκε από τον Eric Brewer, είναι ένας τρόπος διάκρισης του σχεδιασμού ενός κατανεμημένου συστήματος. Σύμφωνα με το παραπάνω θεώρημα, σε ένα κατανεμημένο σύστημα είναι αδύνατο να ικανοποιηθούν ταυτόχρονα τα χαρακτηριστικά: Συνέπεια (Consistency), Διαθεσιμότητα (Availability) και Ανοχή σφαλμάτων κατάτμησης (Partition fault tolerance) (Pubudumal, 2020). Η Εικόνα 3 επεξηγεί το θεώρημα CAP με τη χρήση Venn διαγραμμάτων.



Εικόνα 3: CAP Theorem in Distributed Systems - https://towardsdatascience.com, 2018, Syed Sadat Nazrul

Συνοπτικά, όταν γίνεται λόγος στη συνέπεια γίνεται αναφορά στο βασικό χαρακτηριστικό των κατανεμημένων συστημάτων, την αναπαραγωγή/αντιγραφή (replication) καθώς λειτουργούν σε τμήματα/ομάδες που πρέπει να επικοινωνούν μεταξύ τους για επιτυχημένη συνεργασία. Εφόσον δεν υπάρχει η έννοια μιας καθολικής μνήμης, κάθε κόμβος είναι υπεύθυνος για τη διατήρηση ενός τοπικού αντιγράφου των τμημάτων μνήμης που μοιράζονται ανάμεσα στο σύμπλεγμα των επιλεγμένων κόμβων. Επομένως όταν πραγματοποιείται μια ενημέρωση στο σύστημα (συνήθως προερχόμενη από τους χρήστες του συστήματος), επιβάλλεται να αντικατοπτρίζεται σε όλους τους κόμβους του συστήματος έτσι ώστε όλοι οι χρήστες να εξυπηρετούνται σύμφωνα με τις τελευταίες ενημερώσεις ανεξαρτήτως ποιος κόμβος έχει αναλάβει την εξυπηρέτηση τους. Όταν ένα σύστημα δεν ικανοποιεί ανάλογη συνέπεια ονομάζεται ασυνεπές σύστημα.

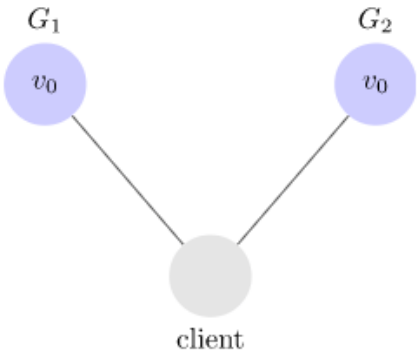
Όσον αφορά τη Διαθεσιμότητα συνηθίζεται να μετριέται σύμφωνα με το χρόνο διαθεσιμότητας/αναμονής (up-time). Πάραυτα, υπάρχει μια βασική διαφορά μεταξύ της γενικότερης διαθεσιμότητας και της διαθεσιμότητας CAP. Εάν ένας κόμβος δεν αποτύχει, σύμφωνα με τον ορισμό της Διαθεσιμότητας CAP συνάγεται πως είναι σε θέση να ανταποκρίνεται σε εισερχόμενα μηνύματα, ακόμα και αν οι απαντήσεις είναι λανθασμένες. Επίσης στο πλαίσιο του θεωρήματος CAP η διαθεσιμότητα του συστήματος προϋποθέτει απεριόριστο χρόνο απόκρισης, η οποία μπορεί να αποτελέσει μια λιγότερο πρακτική τακτική. Γενικότερα, η πραγματικότητα απαιτεί ότι η διαθεσιμότητα πρέπει να μετριέται

για ολόκληρο το σύστημα και όχι για κάθε κόμβο ξεχωριστά καθώς ένα σύστημα με υψηλή διαθεσιμότητα είναι σημαντικότερο και επηρεάζει άμεσα την εμπειρία του χρήστη με το σύστημα και τις προσφερόμενες υπηρεσίες. Ένα διαθέσιμο σύστημα δεν πρέπει να αποτυγχάνει να παρέχει τη ζητούμενη υπηρεσία, ακόμα και αν υπάρχουν κομβόι που έχουν αποτύχει εσωτερικά. Παρατηρείται πως το χαρακτηριστικό της διαθεσιμότητας επιτυγχάνεται ευκολότερα με την αναπαραγωγή των κόμβων.

Επιπροσθέτως, είναι αναπόφευκτο το δίκτυο να αποτυγχάνει περιστασιακά λόγω φυσικών αλλά και λογικών δυσκολιών. Ως αποτέλεσμα των βλαβών του δικτύου, τα δίκτυα χωρίζονται αναπόφευκτα σε πολλές ομάδες. Σύμφωνα με το θεώρημα CAP, η Ανοχή σφαλμάτων της παραπάνω κατάτμησης ορίζεται ως η δυνατότητα να λογοδοτήσει για οποιαδήποτε τυχόν απώλεια μηνύματος μεταξύ τμημάτων. Είναι ιδιαίτερα σημαντικό το σύστημα να είναι σε θέση να λογοδοτήσει για τυχόν σφάλματα κατά τη διαδικασία κατάτμησης.

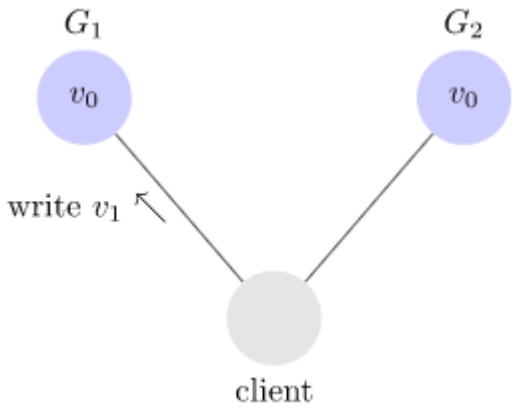
Εφόσον οι εννοείς της συνέπειας, της διαθεσιμότητας και της ανοχής κατάτμησης έχουν οριστεί επαρκώς παραπάνω, σε αυτό το σημείο μπορεί να αποδειχθεί πως ένα σύστημα δεν μπορεί ταυτόχρονα να καλύπτει και τις τρεις ανάγκες.

Έστω, για αντιφατικούς σκοπούς, η υπόθεση πως υπάρχει ένα σύστημα το οποίο είναι συνεπές, διαθέσιμο και ανθεκτικό στην κατάτμηση. Το πρώτο βήμα που πρέπει να υλοποιηθεί είναι ο χωρισμός του συστήματος όπως φαίνεται στην Εικόνα 4 :



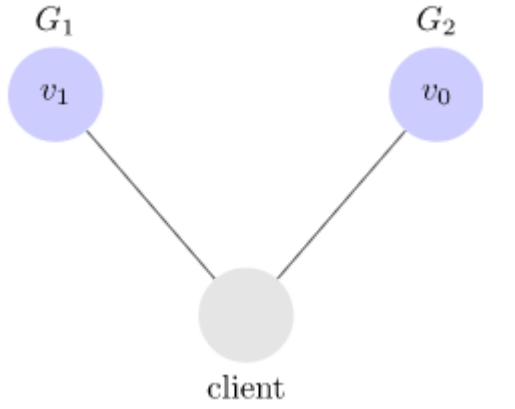
Εικόνα 4: πρώτο βήμα – κατανομή συστήματος σε δύο κόμβους

Στη συνέχεια, προκύπτει το αίτημα του πελάτη να εγγραφεί η πληροφορία v1 στον κόμβο G1 (Εικόνα 5).



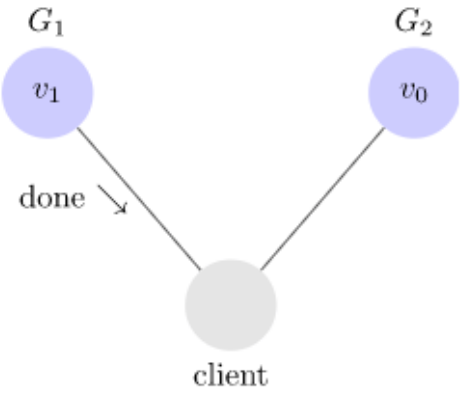
Εικόνα 5: Ο πελάτης αιτείται εγγραφή της πληροφορίας v1 από τον κόμβο G1

Εφόσον πρόκειται για σύστημα με υψηλό ποσοστό διαθεσιμότητας, ο κόμβος G1 πρέπει να ανταποκριθεί (Εικόνα 6).



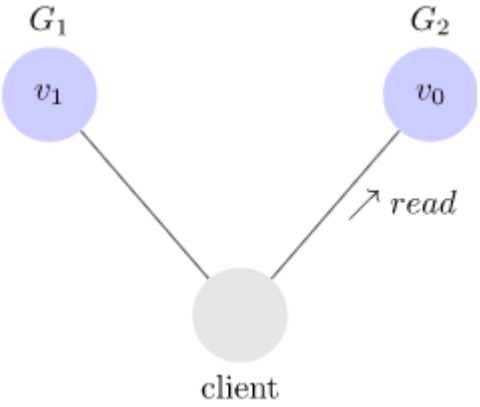
Εικόνα 6: Ο κόμβος G1 κατέχει πλέον την ανανεωμένη πληροφορία v1

Είναι δεδομένο πως το σύστημα είναι κατανεμημένο, ωστόσο, ο κόμβος G1 δεν μπορεί να αναπαράγει τα δεδομένα του στον κόμβο G2 (φάση εκτέλεσης α1) (Εικόνα 7).



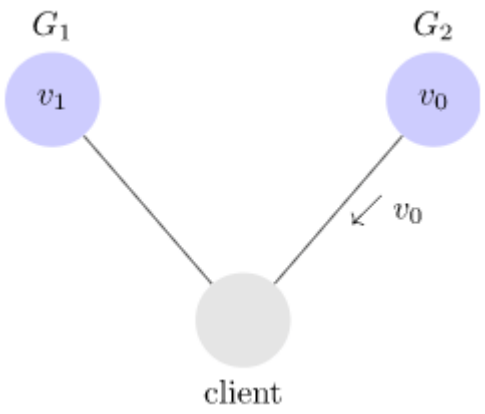
Εικόνα 7: Ο κόμβος G1 ανταποκρίθηκε επιτυχώς αλλά δεν έχει τη δυνατότητα να αναπαράγει την αλλαγή στο γειτονικό του κόμβο G2.

Αμέσως μετά, έστω πως χρήστης πραγματοποιεί μια αίτηση ανάγνωσης για τον κόμβο G2 (Εικόνα 8). Όπως και προηγουμένως, εφόσον πρόκειται για ένα διαθέσιμο σύστημα, ο κόμβος G2 οφείλει να ανταποκριθεί.



Εικόνα 8: Ο χρήστης πραγματοποιεί αίτημα ανάγνωσης προς τον κόμβο G2.

Δεδομένου πως το δίκτυο είναι διαχωρισμένο, ο κόμβος G2 δεν είναι ενημερωμένος με τα δεδομένα του κόμβου G1, επομένως επιστρέφει την τιμή v0 (φάση εκτέλεσης α2) (Εικόνα 9):



Εικόνα 9: Ο κόμβος G2 επιστρέφει την τιμή ν0.

Το γεγονός πως ο κόμβος G2 επιστρέφει την τιμή ν0 στο αίτημα ανάγνωσης του χρήστη, ενώ ο κόμβος G1 έχει λάβει την πιο πρόσφατη τιμή ν1, αποτελεί φαινόμενο ασυνέπειας εντός του συστήματος.

Η υπόθεση ότι υπήρχε ένα συνεπές, διαθέσιμο, ανθεκτικό στο διαχωρισμό σύστημα καταρρίφθηκε μέσω της επίδειξης ότι υπάρχει μια εκτέλεση για οποιοδήποτε τέτοιο σύστημα στο οποίο το σύστημα δεν τηρεί τις προϋποθέσεις συνέπειας όπου απαιτούνται. Συνεπώς, δεν υπάρχει τέτοιο σύστημα (Whittaker, 2018).

2.1.3 Πλεονεκτήματα χρήσης Κατανεμημένων συστημάτων

Πολλές εφαρμογές είναι εκ φύσεως κατανεμημένες, επομένως σε μερικές περιπτώσεις είναι αμέσως επόμενο η επιλογή της κατανεμημένης λογικής. Παραδείγματος χάριν, το διαδίκτυο είναι ένα πολύ γνωστό κατανεμημένο τεχνολογικό σύστημα. Η πρόσβαση στο διαδίκτυο γίνεται μέσω ενός περιηγητή (browser), ο οποίος λειτουργεί μέσω μιας ηλεκτρονικής συσκευής. Συνολικά όλες οι συσκευές που χρησιμοποιούνται για τη σύνδεση στον παγκόσμιο ιστό σχηματίζουν ένα κατανεμημένο σύστημα.

Επιπλέον, η διαθεσιμότητα είναι ένας πολύ σημαντικός παράγοντας που κάνει τη χρήση κατανεμημένων συστημάτων αναγκαία. Πολλές εφαρμογές απαιτούν ανθεκτικότητα κατά των σφαλμάτων που μπορεί να προκληθούν σε μια εφαρμογή που βασίζεται στη λειτουργικότητα ενός κόμβου. Οι σύγχρονες εφαρμογές απαιτούν άμεση διαθεσιμότητα, η οποία αν απουσιάζει μπορεί να υποβιβάσει καθολικά την αξιοπιστία της εφαρμογής και να προκαλέσει τη συνολική αποτυχία της στην αγορά.

Πολλές εφαρμογές δέχονται πολύ μεγάλο φόρτο εργασίας το οποίο πρέπει να επεξεργαστούν, το οποίο είναι αδύνατον να πραγματοποιηθεί σε έναν μοναδικό κόμβο, σε

αποδεκτό χρόνο. Η χρήση πιο ισχυρών πόρων για αυτό τον ένα κόμβο δεν μπορεί να διορθώσει αυτό το πρόβλημα. Αρκεί να αναλογιστεί κάποιος το φόρτο εργασίας που δέχονται οι μηχανές αναζήτησης κάθε στιγμή σε παγκόσμια κλίμακα για να γίνει αυτονόητο πως η κατανομή του φόρτου είναι απαραίτητη.

Τέλος, ένα κατανεμημένο σύστημα υπερτερεί σημαντικά στην απόδοση και ταχύτητα που μπορεί να προσφέρει. Υπάρχουν διάφορες εφαρμογές, που βασίζονται σε υψηλές ταχύτητες, όπως για παράδειγμα οι υπηρεσίες παροχής video streaming. Η ύπαρξη πολλαπλών κέντρων δεδομένων (data centres) με τη χρήση κατανεμημένων τεχνικών μπορεί να προσφέρει την επιθυμητή απόδοση για τέτοιου είδους υπηρεσίες.

2.1.4 Μειονεκτήματα Κατανεμημένων Συστημάτων

Ενώ η τεχνικές των κατανεμημένων συστημάτων έχουν σημαντικά πλεονεκτήματα συγκριτικά με παλαιότερα συστήματα, το γεγονός πως η εκτέλεση της εφαρμογής συμβαίνει παράλληλα σε διαφορετικά υπολογιστικά συστήματα μπορεί να δημιουργήσει κάποιες προκλήσεις. Κάποιες από τις βασικές δυσκολίες - προκλήσεις που παρατηρούνται είναι οι εξής (Anthony, 2016):

- Δομική, επικοινωνιακή, διοικητική πολυπλοκότητα μεταξύ των κατανεμημένων κόμβων. Η ανταλλαγή αποτελεσμάτων και η επικοινωνία μεταξύ των ξεχωριστών συστημάτων αυξάνει τη γενικότερη πολυπλοκότητα, κάνοντας τη συνολική εφαρμογή δυσκολότερα προβλέψιμη και ελέγξιμη μέσω δοκίμων και τεστ.
- Η χρήση τουλάχιστον ενός διαχειριστή (συνήθως κάποια τεχνολογία Load Balancing ή ένας επιπλέον κόμβος) ο οποίος είναι υπεύθυνος για την ίση κατανομή των αιτημάτων ή των δεδομένων είναι απαραίτητη για την αποφυγή ανεπιθύμητων αντιγράφων. Ακόμα, ο διαχειριστής πρέπει να διατηρεί όλους τους κόμβους ενήμερους κάθε στιγμή.
- Υπάρχει η πιθανότητα να προκύψουν δυναμικές αλλαγές στο σχεδιασμό των πόρων του συστήματος αλλά και στο φόρτο εργασίας που δέχεται το σύστημα. Αυτό είναι ικανό να δημιουργήσει απότομες αλλαγές στη διαθεσιμότητα και την απόδοση.
- Οι πόροι του συστήματος μπορούν να είναι μεταβαλλόμενοι σε όλο το σύστημα, αλλά η κάθε διαδικασία πρέπει να βρίσκεται σε θέση να εξυπηρετείται on demand

(κατά παραγγελία) διαρκώς. Επομένως η εύρεση των πόρων πρέπει να πραγματοποιείται μέσω αυτόματης εύρεσης των διαθέσιμων πόρων.

- Πολλαπλές διαδικασίες - κόμβοι έχουν πρόσβαση σε κοινόχρηστους πόρους. Η πρόσβαση στους πόρους θα πρέπει να ρυθμίζεται μέσω ειδικών μηχανισμών με σκοπό να διασφαλιστεί η συνέπεια μεταξύ των διαδικασιών. Το παραπάνω σημαίνει πως μπορεί να χρειαστεί η στειροποίηση των ενημερώσεων ώστε να διασφαλιστεί η απουσία παρεμβολών και άλλων προσβάσεων.

2.1.5 Τύποι κατανομής

Τα κατανεμημένα συστήματα μπορούν να διαχωριστούν σε διάφορες κατηγορίες κατανομής ανάλογα με τις πτυχές του συστήματος που διανέμονται και τον τρόπο με τον οποίο επιτυγχάνεται η διανομή. Οι πτυχές που μπορούν να διανεμηθούν είναι οι εξής (Anthony, 2016):

- Αρχεία: Πρόκειται για τις αρχικές μορφές κατανομής ανάμεσα σε συστήματα και πλέον θεωρείται κοινή πρακτική. Τα αρχεία μπορούν να εξαπλωθούν σε όλο το σύστημα με σκοπό διαμοιραστεί το φορτίο σε πολλές διεργασίες του συστήματος. Ακόμα., μια κοινή πρακτική είναι η δημιουργία αντιγράφων των αρχείων σε πολλαπλές διεργασίες του διακομιστή ώστε να διασφαλιστεί η προσβασιμότητα και η ευρωστία στην περίπτωση που ένα μεμονωμένο αντίγραφο του αρχείου καταστραφεί.
- Λειτουργικό σύστημα: Ένα λειτουργικό σύστημα είναι δυνατόν να διαμοιραστεί σε πολλαπλές συσκευές ηλεκτρονικού υπολογιστή. Ο στόχος της κατανομής του λειτουργικού συστήματος είναι παρέχει μια αφηρημένη εικόνα ενός υπολογιστικού συστήματος οπού η κάθε διεργασία εκτελείται χωρίς να υπάρχει άμεση σύνδεση με ένα κεντρικό υπολογιστή. Ουσιαστικά η κάθε εντολή ή διεργασία πραγματοποιούνται σε ένα κεντρικό υπολογιστή, όμως οι λεπτομέρειες καλύπτονται κατά τη διαδικασία. Το ενδιάμεσο λογισμικό (middleware) επιτυγχάνει ανάλογα αποτελέσματα σε ένα υψηλότερο επίπεδο. Θα γίνει αναλυτικότερη αναφορά στον τρόπο λειτουργίας του ενδιάμεσου λογισμικού παρακάτω.

- Φόρτο Εργασίας: Η διασπορά των εργασιών είναι μια από τις πιο σύνηθες πρακτικές στα κατανεμημένα συστήματα, καθώς με αυτό τον τρόπο οι εργασίες μπορούν να εξαπλωθούν δυναμικά σε όλους τους διαθέσιμους πόρους ώστε να διαμοιραστεί το αρχικό φορτίο. Ο κύριος στόχος είναι η βελτίωση της συνολικής απόδοσης του συστήματος, με το να μην επιβαρύνονται ήδη επιβαρυμένοι πόροι κατευθύνοντας το φορτίο εργασίας σε πόρους που χρησιμοποιούνται πιο σπάνια ή είναι πιο άμεσα διαθέσιμοι. Παρόλα τα θετικά που προσφέρει η κατανομή του φόρτου εργασίας, πρέπει να διασφαλιστεί πως τέτοια σχήματα κατανομή παραμένουν σταθερά υπό όλες τις συνθήκες και πως τα πλεονεκτήματα που προσφέρονται υπερβαίνουν τα αρνητικά, όπως η δυσκολία που μπορεί να προκύψει όσον αφορά την παρακολούθηση, διαχείριση και μεταφορές εργασιών του συστήματος.
- Δεδομένα: Η χρήση κατανεμημένων βάσεων δεδομένων επιτρέπουν τη γεωγραφική διασπορά των δεδομένων με τέτοιο τρόπο ώστε να βρίσκονται γεωγραφικά πλησιέστερα στους χρήστες που τα χρειάζονται. Με αυτό τον τρόπο επιτυγχάνεται υψηλότερη ταχύτητα πρόσβασης για την πλειοψηφία των χρηστών. Με την παραπάνω τεχνική κατανομής των δεδομένων, κατανέμεται παράλληλα και το φορτίο εργασίας σε διάφορους διακομιστές (servers) και όπως αναφέρθηκε ήδη με αυτό τον τρόπο επιτυγχάνεται αυξημένο ποσοστό διαθεσιμότητας και γενικότερης ανθεκτικότητας του συστήματος.
- Παράλληλα, υπηρεσίες υποστήριξης της λειτουργίας των κατανεμημένων συστημάτων (συστήματα ειδοποιήσεων, υπηρεσίες ονομασιών, διεργασίες διαχείρισης συστημάτων) μπορούν οι ίδιες να κατανεμηθούν ώστε να βελτιωθεί η αποτελεσματικότητα τους.
- Στην πλειονότητα των περιπτώσεων, εφαρμόζεται ένας συνδυασμός των παραπάνω τύπων κατανομής. Οι εφαρμογές χωρίζονται σε πολλαπλά ¨κομμάτια¨ και το κάθε στοιχείο κατανέμεται εσωτερικά στο σύστημα ανάλογα τις απαιτήσεις. Συνήθως υπάρχει ένα στοιχείο το οποίο είναι υπεύθυνο για τη διεπαφή του χρήστη αλλά και κάποιο ποσοστό τοπικής επεξεργασίας και αποθήκευσης δεδομένων. Ακόμα, υπάρχει τουλάχιστον ένα user-remote μέρος του συστήματος (user-remote parts) που είναι τοποθετημένο γειτονικά σε κοινόχρηστα δεδομένα (shared data) και άλλους διαθέσιμους πόρους.

2.1.6 Μέτρηση ποιότητας των κατανεμημένων συστημάτων

Οι απαιτήσεις κατανεμημένων εφαρμογών εμπίπτουν σε δύο κατηγορίες: τις λειτουργικές απαιτήσεις (functional–behavior requirements) της εφαρμογής και τις μη λειτουργικές απαιτήσεις (nonfunctional requirements) όπου σχετίζονται με τη συνολική ποιότητα της εφαρμογής. Οι μη-λειτουργικές απαιτήσεις είναι πιο περίπλοκες στη μέτρηση τους καθώς είναι δυσκολότερα ελέγξιμες. Κατά κύριο λόγο οι λειτουργικές και μη-λειτουργικές απαιτήσεις είναι άμεσα συνδεδεμένες, καθώς σχεδόν όλα τα λειτουργικά μέρη μιας εφαρμογής συμβάλλουν στην επίτευξη μη-λειτουργικών πτυχών, οι οποίες με τη σειρά τους επιδρούν στην πληρότητα των λειτουργικών απαιτήσεων. Επομένως, όλα τα στάδια του σχεδιασμού και της ανάπτυξης ενός κατανεμημένου συστήματος θα πρέπει να λαμβάνουν υπ’ όψιν και να δίνεται η πρέπουσα σημασία στις μη λειτουργικές ανάγκες της εφαρμογής

Οι λειτουργικές απαιτήσεις είναι συγκεκριμένες και σχετίζονται με τη συμπεριφορά ή τα αναμενόμενα αποτελέσματα στο πλαίσιο της κάθε εφαρμογής. Είναι σημαντικό να έχουν καθοριστεί αναλυτικά, σε συνδυασμό με σαφής οδηγίες δοκίμων, έτσι ώστε να είναι δυνατόν να επιβεβαιωθεί η σωστή ερμηνεία των προϋποθέσεων αλλά και πως το τελικό αποτέλεσμα είναι το επιθυμητό.

Αντιθέτως, οι μη λειτουργικές απαιτήσεις είναι λιγότερο σαφώς καθορισμένες και η ερμηνεία τους είναι ανοιχτή σε κάποιο βαθμό. Τυπικές μη-λειτουργικές απαιτήσεις αποτελούν ή επεκτασιμότητα (scalability), η διαθεσιμότητα (availability), η ανθεκτικότητα (robustness), η ταχύτητα ανταπόκρισης (responsiveness) και η διαφάνεια (transparency). Ενώ, όλες αυτές οι απαιτήσεις είναι πολύ σημαντικές στη μέτρηση ποιότητας ενός κατανεμημένου συστήματος, η διαφάνεια θεωρείται η πιο σημαντική προϋπόθεση για την επιτυχία της εφαρμογής.

2.2 Επικοινωνία

Γίνεται γρηγορά κατανοητό πως η επικοινωνία μεταξύ των κόμβων ενός κατανεμημένου συστήματος είναι ένα σημαντικό μέρος του σχεδιασμού μιας κατανεμημένης εφαρμογής. Οι κομβόι απαιτούν τον ανάλογο συντονισμό μέσα από την άμεση ή έμμεση επικοινωνία μεταξύ τους. Γενικά, τα κατανεμημένα συστήματα πρέπει να σχεδιάζονται έτσι ώστε να

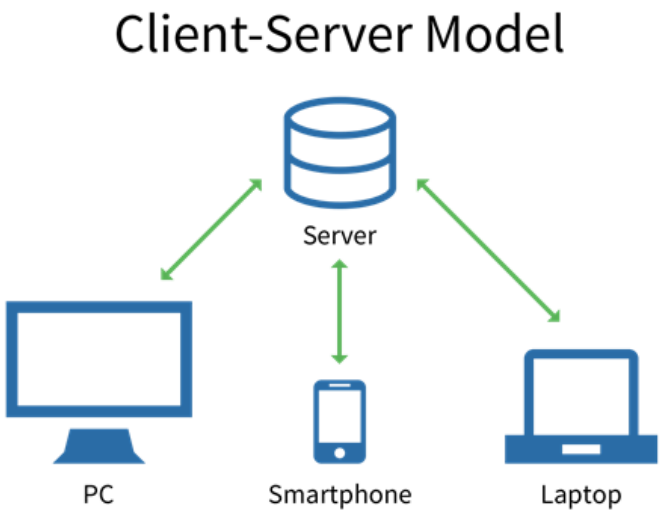
είναι ανεκτικά σε σφάλματα, το οποίο σημαίνει πως η λειτουργία τους συνεχίζεται παρά την ύπαρξη ενός ή παραπάνω σφαλμάτων.

2.3 Αρχιτεκτονικά Μοντέλα Κατανεμημένων Συστημάτων

Καθώς στην εποχή μας συνεχώς αυξάνονται οι τεχνολογικές απαιτήσεις αλλά και διευρύνονται σε όλους τους τομείς, τα κατανεμημένα συστήματα και η διαχείριση ανάπτυξης τους γίνονται πιο περίπλοκα με αποτέλεσμα ο διαχωρισμός τους να γίνεται επιτακτικός. Ονομάζουμε ως αρχιτεκτονικά μοντέλα τους τρόπους οργάνωσης των κατανεμημένων τμημάτων αλλά και με βάση τους διάφορους τρόπους δόμησης των μεταξύ τους συσχετίσεων (Faisandier, Roedler, & Adcock, 2021). Βασικός στόχος της κατηγοριοποίησης των αρχιτεκτονικών μοντέλων είναι η ευκολότερη κατανόηση του κάθε συστήματος, η οργανωμένη ανάπτυξη του, η επαναχρησιμοποίηση των συστατικών του αλλά και για την εξέλιξη και συντήρηση του συστήματος. Το μοντέλο πελάτη-εξυπηρετητή και το μοντέλο ομότιμων είναι οι δύο κύριες αρχιτεκτονικές έννοιες για κατανεμημένα συστήματα.

2.3.1 Μοντέλο Πελάτη - Εξυπηρετητή (Client-Server Model)

Το μοντέλο Πελάτη – Εξυπηρετητή περιγράφει τη διαδικασία όπου ο διακομιστής (server) παρέχει πρόσβαση σε πόρους και υπηρεσίες σε έναν ή περισσότερους πελάτες. Web servers, διακομιστές αλληλογραφίας και αρχείων αποτελούν παραδείγματα διακομιστών που έχουν υιοθετήσει το μοντέλο πελάτη – εξυπηρετητή (Oluwatosin, 2014). Ως πελάτης μπορεί να θεωρηθεί οποιαδήποτε συσκευή παρέχεται με πόρους από τους εξυπηρετητές, όπως υπολογιστές γραφείου, φορητοί υπολογιστές, tablet, smartphone και οποιαδήποτε συσκευή IOT συνδέεται με κάποιον διακομιστή με σκοπό τη χρήση των πόρων του. Οι περισσότεροι διακομιστές εγκαθιδρύουν σχέσεις one-to-many με τους πελάτες, το οποίο σημαίνει πως ένας διακομιστής έχει τη δυνατότητα να εξυπηρετεί πολλούς πελάτες ταυτόχρονα, όπως γίνεται εμφανές στην Εικόνα 10.



Εικόνα 10: Ο διακομιστής μπορεί να εξυπηρετεί διάφορα είδη πελατών ταυτόχρονα

Όταν ένας χρήστης/πελάτης αποστέλλει αίτημα σύνδεσης στο διακομιστή, ο διακομιστής έχει την επιλογή αποδοχής ή απόρριψης του αιτήματος. Εάν η σύνδεση γίνει αποδεκτή, ο διακομιστής αναλαμβάνει τη δημιουργία και διατήρηση μιας σύνδεσης με το χρήστη μέσω του κατάλληλα επιλεγμένου πρωτοκόλλου. Παραδείγματος χάριν, οι διακομιστές ηλεκτρονικής αλληλογραφίας συνήθως χρησιμοποιούν το πρωτόκολλο SMTP έτσι ώστε να προσφέρουν την υπηρεσία αποστολής μηνυμάτων στους χρήστες τους. Το συγκεκριμένο πρωτόκολλο θα απαιτήσει τον ανάλογο έλεγχο ταυτότητας (συνήθως πρόκειται για τη διεύθυνση ηλεκτρονικού ταχυδρομείου και τον κωδικό του χρήστη) και εφόσον ο χρήστης τακτοποιηθεί επιτυχώς, ο διακομιστής θα προβεί στη διεκπεραίωση των επιθυμητών ενεργειών, όπως η αποστολή ενός μηνύματος στο παρόν παράδειγμα (Zhang, 2013).

Παρόλο που οι διαδικτυακοί διακομιστές έχουν τη δυνατότητα να συνδέονται με πολλούς πελάτες ταυτόχρονα, υπάρχει ένα όριο στην κίνηση που μπορεί να χειριστεί μια φυσική μηχανή. Ως αποτέλεσμα, οι πιο επιφανείς διαδικτυακές υπηρεσίες υιοθετούν την τεχνική του κατανεμημένου υπολογισμού (distributed computing) με σκοπό τη διανομή των χρηστών σε πολλούς φυσικούς διακομιστές. Στις περισσότερες περιπτώσεις δεν έχει σημασία ποιος διακομιστής εξυπηρετεί κάθε χρήστη, καθώς όλοι οι διακομιστές παρέχουν την ιδιά υπηρεσία (Christensson , 2016).

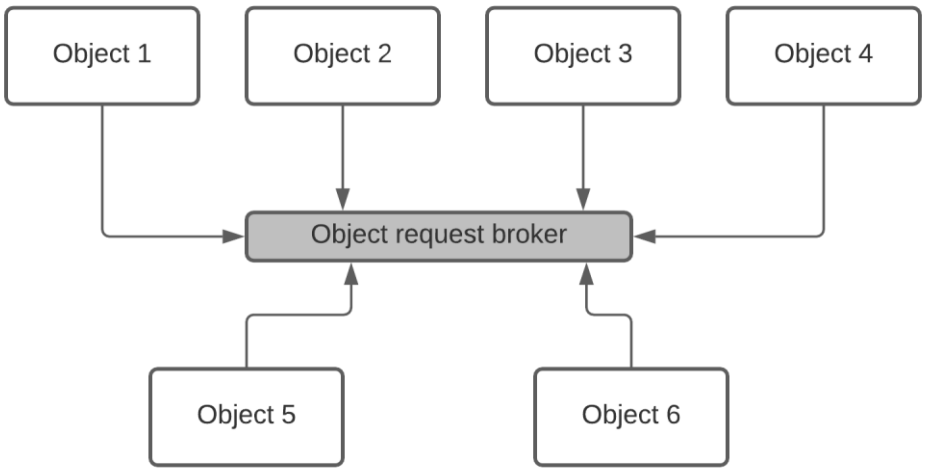
Το παραπάνω μοντέλο προσφέρει πλεονεκτήματα όπως τη δυνατότητα κατανομής της απαιτούμενης επεξεργασίας σε διαφορά υπολογιστικά συστήματα. Επιπλέον επιτρέπει ευκολότερη κοινή χρήση πόρων μεταξύ πελατών και διακομιστών. Ακόμα, μειώνεται σημαντικά το φόρτο εργασίας για την πλευρά των πελατών, καθώς διαδικασίες όπως η αναπαραγωγή δεδομένων γίνεται από τους διακομιστές, οι οποίοι είναι υπεύθυνοι για την αναπαραγωγή των δεδομένων αποθηκεύοντας τα δεδομένα τοπικά.

Καθώς όλο και περισσότερες επιχειρήσεις μετακινούνται στο cloud (cloud computing, υπολογιστικό νέφος), η αρχιτεκτονική πελάτη – εξυπηρετητή γίνεται πιο διαδεδομένη. Διάφορες μορφές υπηρεσιών, όπως τα κατανεμημένα συστήματα και το cloud computing, αυξάνονται σε δημοτικότητα.

2.3.2 Αρχιτεκτονική κατανεμημένων αντικειμένων (Distributed object architecture)

Κατά την ανάλυση των διαφορετικών αρχιτεκτονικών των κατανεμημένων συστημάτων είναι σημαντική η αναφορά στην αρχιτεκτονική κατανεμημένων αντικειμένων. Στην παρούσα αρχιτεκτονική δεν πραγματοποιείται διάκριση μεταξύ διακομιστών και πελατών. Συγκεκριμένα, οποιαδήποτε πλευρά ή οντότητα του συστήματος έχει τη δυνατότητα να παρέχει τις ζητούμενες υπηρεσίες στα υπόλοιπα αντικείμενα και αντίστοιχα να χρησιμοποιεί υπηρεσίες άλλων αντικειμένων (Sommerville, 2006).

Ως κατανεμημένα αντικείμενα ορίζονται τα αντικείμενα όπου συνθέτουν το σύστημα ως σύνολο και εκτελούνται σε διαφορετικά περιβάλλοντα εκτέλεσης. Σε κάθε δεδομένη χρονική στιγμή κάθε αντικείμενο οφείλει να είναι σε θέση να εκτελέσει επιτυχώς μια συγκεκριμένη εργασία που του έχει ανατεθεί, επομένως το κάθε αντικείμενο θα πρέπει να έχει πρόσβαση στους απαραίτητους πόρους όπως αρχεία, βάσεις δεδομένων, κ.α. για την εκτέλεση της εργασίας. Παράλληλα, το κάθε αντικείμενο στο σύστημα είναι απαραίτητο να έχει τη γνώση όσον αφορά τον κώδικα έτσι ώστε να έχει τη δυνατότητα να γνωρίζει πως εκτελείται η κάθε διεργασία. Χρειάζεται επίσης να σημειωθεί πως για την επικοινωνία των αντικειμένων χρησιμοποιείται ένα ενδιάμεσο λογισμικό (Middleware System), το οποίο ονομάζεται object request broker (Distributed Objects, 2000). Ένα τυπικό σχήμα της παραπάνω αρχιτεκτονικής εμφανίζεται στην Εικόνα 11.



Εικόνα 11: Τυπικό σχήμα αρχιτεκτονικής κατανεμημένων αντικειμένων

Γνωστές εφαρμογές της αρχιτεκτονικής κατανεμημένων αντικειμένων αποτελούν οι:

- Αρχιτεκτονική CORBA (Common Object Request Broker Architecture)
- Remote Method Invocation (RMI)
- Αρχιτεκτονική DCOM (Distributed Component Model)
- Υπηρεσίες Ιστού (Web Services)
- Κινητοί Πράκτορες (mobile agents)
- Πλέγμα (Grid)

Μεταξύ των θετικών επιπτώσεων μπορεί να αναφερθεί η δυνατότητα να σχεδιαστούν σχετικά αργότερα οι προσφερόμενες υπηρεσίες του συστήματος από το σχεδιαστή του συστήματος. Επιπλέον, πρόκειται για ένα είδος αρχιτεκτονικής με ιδιαίτερα ανοιχτό σύστημα, το οποίο επιτρέπει την προσθήκη νέων πόρων ανάλογα με τις ανάγκες της κάθε χρονικής στιγμής. Κατά συνέπεια, το σύστημα είναι σημαντικά ευέλικτο και εύκολα επεκτάσιμο, χαρακτηριστικά πολύ χρήσιμα για ένα κατανεμημένο σύστημα. Εκτός από τα παραπάνω, η δυναμική επαναδιαμόρφωση ενός συστήματος με την παρούσα αρχιτεκτονική είναι δυνατή μέσω της μετακίνησης των αντικειμένων σε όλο το δίκτυο αντίστοιχα (Comparing Service-Oriented and Distributed Object Architectures, 2005).

2.4 Διαδικασίες – Νήματα

Κατά την ανάπτυξη και το σχεδιασμό κατανεμημένων εφαρμογών είναι ιδιαίτερα σημαντικό να είναι ξεκάθαρες οι διαφορές μεταξύ όρων όπως «πρόγραμμα», «διαδικασία» και «νήμα». Αυτού του είδους ο διαχωρισμός είναι κρίσιμος καθώς αποτελεί τη βάση κάθε εφαρμογής και ιδιαίτερα τεχνικών παραλληλισμού όπως η πολλαπλή επεξεργασία ή η χρήση πολλαπλών νημάτων, οι οποίες αναλύονται αργότερα.

2.4.1 Πρόγραμμα (Program)

Αρχικά είναι σημαντικό να οριστεί ο όρος «πρόγραμμα»:

«Ένα πρόγραμμα είναι μια λίστα με τις οδηγίες μαζί με τις πληροφορίες δομής και ακολουθίας για τον έλεγχο της σειράς κατά την οποία πραγματοποιούνται οι οδηγίες» (Anthony, 2016, σ. 22)

Από το παραπάνω ορισμό γίνεται κατανοητό πως υπάρχουν διάφοροι τύποι προγραμμάτων, συμπεριλαμβανομένων βασικών προγραμμάτων τα οποία βοηθούν τη λειτουργία του υπολογιστικού συστήματος και αποτελούν μέρος του λειτουργικού συστήματος αλλά και άλλα προγράμματα που σκοπεύουν στην εκπλήρωση μιας και μοναδικής συγκεκριμένης εργασίας. Αυτά τα εστιασμένα σε εργασίες προγράμματα, συχνά γνωστά ως "εφαρμογές", μπορεί να περιλαμβάνουν εργαλεία όπως επεξεργασία κειμένου, περιήγηση στο διαδίκτυο και αποστολή μηνύματος μέσω ηλεκτρονικού ταχυδρομείου σε άλλο υπολογιστή.

Τα προγράμματα συνήθως αποθηκεύονται στο δίσκο ή σε σταθερή μνήμη σε μορφή που μπορεί να εκτελεστεί από το υπολογιστικό σύστημα που βρίσκεται. Πριν από την εκτέλεση τους, κατασκευάζονται χρησιμοποιώντας οδηγίες που ενσωματώνουν λογική, χειρισμό δεδομένων και συσκευών, επανάληψη και αλληλεπίδραση χρηστών σε μια γλώσσα προγραμματισμού όπως C, Lisp, Pascal ή πολλές άλλες. Το τελικό αποτέλεσμα είναι ένα αρχείο κειμένου κώδικα που μεταγλωττίζεται σε δυαδική μορφή (μονάδες και μηδενικά) για να εκτελεστεί στα κατώτερα επίπεδα του υπολογιστή. Ένας άλλος τύπος προγράμματος ονομάζεται "ερμηνευμένος" (interpreted) και αντί να μεταγλωττιστεί εκ των προτέρων για να εκτελεστεί, ερμηνεύεται σε εκτελέσιμο κώδικα τη στιγμή που εκτελείται. Μερικές κοινές, τυπικά ερμηνευμένες γλώσσες προγραμματισμού, είναι οι Python, PHP, JavaScript και Ruby.

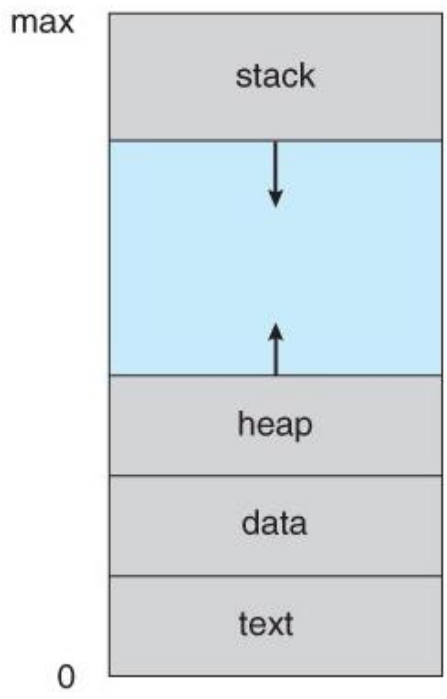
Όποιος και αν είναι ο τύπος προγράμματος που επιλέχθηκε, το τελικό αποτέλεσμα παραμένει το ίδιο κατά την εκτέλεση ενός προγράμματος. Το πρόγραμμα φορτώνεται στη μνήμη σε δυαδική μορφή για να μπορεί να είναι συμβατό με τις δυνατότητες της κεντρικής μονάδας επεξεργασίας του υπολογιστή (CPU), η οποία έχει τη δυνατότητα ανάγνωσης οδηγιών μόνο δυαδικής μορφής, επομένως το πρόγραμμα οφείλει να εκτελείται σε αυτή τη μορφή (Rochkind, 2004).

2.4.2 Διαδικασίες (Processes)

Η διερεύνηση των κατανεμημένων συστημάτων προϋποθέτει την εννοιολογική

του προσέγγιση των διαδικασιών. Αν επιχειρήσουμε να δώσουμε έναν περιεκτικό ορισμό των διαδικασιών ως οντότητα στο πλαίσιο της ανάπτυξης ενός κατανεμημένου συστήματος, θα λέγαμε πως μια διαδικασία αναφέρεται σε ένα σύνολο οδηγιών που επεξεργάζονται επί του παρόντος από τον επεξεργαστή του υπολογιστή.

Μια σημαντική διαφορά μεταξύ του προγράμματος και μιας διαδικασίας είναι πως το ίδιο το πρόγραμμα είναι συλλογή εντολών, οι οποίες περιγράφουν αναλυτικά πως να υλοποιηθεί το κάθε βήμα και εντολή ενώ όταν δημιουργείται μια διαδικασία, πραγματοποιεί τις οδηγίες του σχετικού προγράμματος. Κάθε διαδικασία έχει πολλά στοιχεία που της επιτρέπουν να εκτελέσει τις επιδιωκόμενες εργασίες της. Η στοίβα και ο σωρός, ειδικότερα, χρησιμοποιούνται για τοπική μεταβλητή και δυναμική κατανομή μνήμης, αντίστοιχα. Ορισμένοι βασικοί πόροι που χρειάζεται κάθε διαδικασία είναι καταχωρητές, ένας μετρητής προγράμματος, ο σωρός και μια στοίβα, όπως παρουσιάζεται στην Εικόνα 12 (Imdad, 2015).



Εικόνα 12: Διαδικασία στη μνήμη

Συνοπτικά, οι καταχωρητές (registers) είναι περιοχές αποθήκευσης δεδομένων που είναι ενσωματωμένες στη CPU. Ένας καταχωρητής μπορεί να αποθηκεύσει μια οδηγία, μια θέση αποθήκευσης ή οποιοδήποτε άλλο τύπο δεδομένων που απαιτεί η λειτουργία.

Ο μετρητής προγράμματος (program counter) ή γνωστός και ως δείκτης εντολών (instruction pointer), όπως αναφέρθηκε και κατά τον ορισμό των προγραμμάτων παραπάνω, παρακολουθεί πού βρίσκεται ένας υπολογιστής στην ακολουθία του προγράμματος.

Ως στοίβα ονομάζεται μια δομή δεδομένων που περιέχει πληροφορίες σχετικά με τις ενεργές υπορουτίνες ενός προγράμματος υπολογιστή και χρησιμεύει ως χώρος μηδενισμού για τη διαδικασία. Διαφέρει από τη δυναμικά κατανεμημένη μνήμη για τη διαδικασία γνωστή ως "σωρός".

Ένα μεμονωμένο πρόγραμμα μπορεί να έχει πολυάριθμες λειτουργίες και παράλληλα εκτελέσιμες εργασίες και κάθε παρουσία αυτού του τρέχοντος προγράμματος είναι μια διαδικασία. Επειδή κάθε διαδικασία έχει το δικό της χώρο διευθύνσεων μνήμης, μπορεί να εκτελείται ανεξάρτητα και διαχωρίζεται από άλλες διεργασίες. Δεν έχει άμεση πρόσβαση σε κοινόχρηστα δεδομένα σε άλλες διαδικασίες. Η μετάβαση μεταξύ διαδικασιών απαιτεί

ένα σχετικά σύντομο χρονικό διάστημα καθώς οι καταχωρητές, οι χάρτες μνήμης και άλλοι πόροι πρέπει να αποθηκευτούν και να φορτωθούν.

Αυτή η ανεξαρτησία της διαδικασίας είναι απαραίτητη επειδή το λειτουργικό σύστημα καταβάλλει κάθε δυνατή προσπάθεια για να διαχωρίσει τις διαδικασίες έτσι ώστε ένα πρόβλημα με το ένα να μην μολύνει ή να διαταράξει το άλλο.

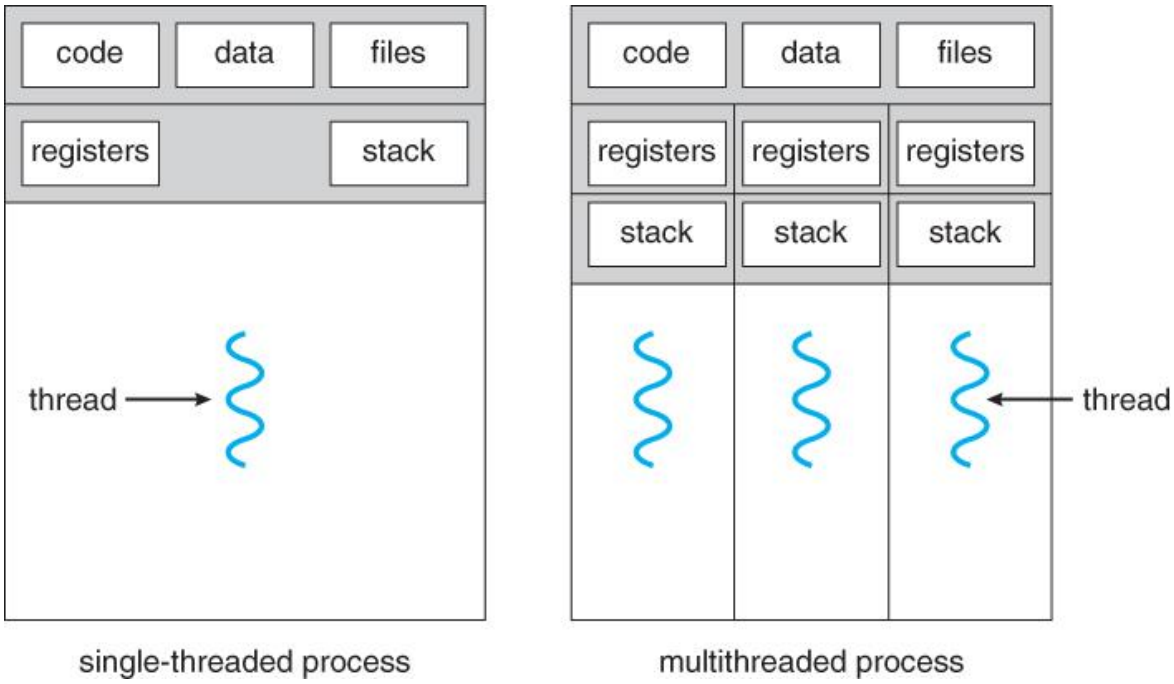
2.4.3 Νήματα (Threads)

Ως νήμα ορίζεται μια αυτοτελής συλλογή τιμών για τους καταχωρητές του επεξεργαστή (αναφορά σε συστήματα ενός πυρήνα). Δεδομένου ότι περιλαμβάνει τον μετρητή προγράμματος (Program Counter ή Instruction Pointer), ελέγχει ποιες εργασίες εκτελούνται με ποια σειρά. Περιλαμβάνει επίσης το δείκτη στοίβας, ο οποίος πρέπει να δείχνει ένα ξεχωριστό τμήμα μνήμης για κάθε νήμα, διαφορετικά θα παρεμβαίνει μεταξύ τους.

Τα νήματα είναι η μονάδα λογισμικού που επηρεάζεται από τη ροή ελέγχου (control flow) (function call, βρόχος επαναλήψεων, Goto εντολές), επειδή αυτές οι οδηγίες λειτουργούν στον μετρητή προγράμματος, ο οποίος ανήκει σε ένα συγκεκριμένο νήμα. Τα νήματα προγραμματίζονται συχνά χρησιμοποιώντας κάποια μέθοδο προτεραιότητας (ενώ παράλληλα είναι δυνατόν να αναπτυχθεί ένα σύστημα με ένα νήμα ανά πυρήνα επεξεργαστή, στην οποία περίπτωση κάθε νήμα λειτουργεί συνεχώς και δεν απαιτείται προγραμματισμός μεταξύ τους).

Επειδή έχουν τη δική τους στοίβα αλλά μπορούν να έχουν πρόσβαση σε κοινόχρηστα δεδομένα, τα νήματα συχνά αναφέρονται ως ελαφρές διαδικασίες. Επειδή τα νήματα μοιράζονται τον ίδιο χώρο διευθύνσεων με τη διαδικασία και άλλα νήματα μέσα στη διαδικασία, το λειτουργικό κόστος της επικοινωνίας νημάτων είναι ελάχιστο, κάτι που είναι πλεονεκτικό. Το μειονέκτημα είναι ότι ένα πρόβλημα με ένα νήμα σε μια διαδικασία θα επηρεάσει σίγουρα άλλα νήματα και τη βιωσιμότητα της ίδιας της διαδικασίας. (Bauer, 2017)

Το παρακάτω σχήμα της εικόνας 13 γίνεται εμφανές ο τρόπος λειτουργίας των διαδικασιών και των νημάτων:



Εικόνα 13:Διαδικασίες μονών και πολλαπλών νημάτων διαδικασίες -πηγή: UIC Computer Science

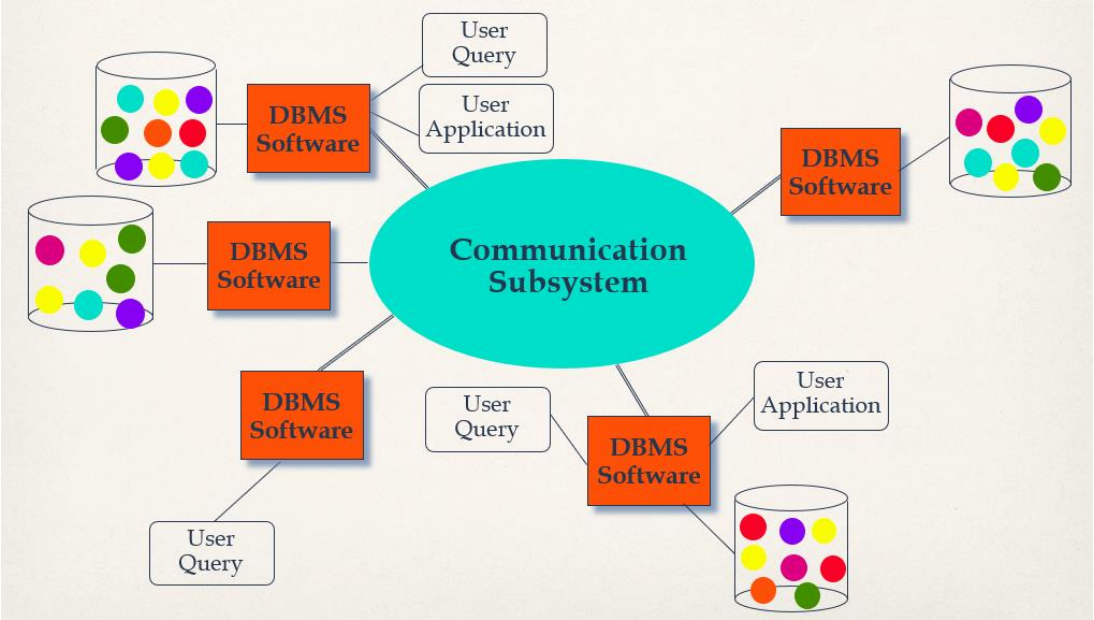
3. Κατανεμημένη Διαχείριση δεδομένων

Ένα από τα θεμελιώδη ερωτήματα στο χώρο των κατανεμημένων συστημάτων είναι ποιο μέρος του συστήματος πρέπει να κατανεμηθεί. Όπως ήδη προαναφέρθηκε μπορεί να υπάρξει κατανομή κατά τη λογική της επεξεργασίας της εφαρμογής. Πράγματι, κατά τον ορισμό του Tanenbaum και van Steen (2002) (Steen & Tanenbaum, 2002) για τα κατανεμημένα υπολογιστικά συστήματα, συνήθως απαιτείται η κατανομή της λογικής επεξεργασίας ή των διαθέσιμων μονάδων επεξεργασίας. Μια άλλη επιλογή διανομής βασίζεται στη λειτουργία. Πολλές από τις λειτουργίες ενός υπολογιστικού συστήματος είναι δυνατόν να ανατεθούν σε διάφορα μέρη λογισμικού ή υλικού (hardware). Ένας ακόμα εναλλακτικός τρόπος διανομή βασίζεται στην κατανομή των δεδομένων, το οποίο είναι το κύριο θέμα που θα αναλυθεί σε αυτή την ενότητα. Τα δεδομένα που χρησιμοποιούνται από πολλές εφαρμογές ενδέχεται να αποστέλλονται σε διάφορες τοποθεσίες για επεξεργασία. Τέλος, ο έλεγχος ενός συστήματος είναι δυνατόν να κατανεμηθεί. Αντί να πραγματοποιείται ο έλεγχος από ένα μοναδικό υπολογιστικό σύστημα, η εκτέλεση διαφορετικών εργασιών μπορεί να εξαπλωθεί (Özsu & Valduriez, 2011).

Από μια ευρύτερη οπτική, το κύριο κίνητρο για την όλο και αυξανόμενη τάση προς την κατανομή των συστημάτων αποτελεί η ανάγκη για βελτίωση της ικανότητας μας να αντιμετωπίζουμε τις σημερινές μεγάλης κλίμακας δυσκολίες διαχείρισης δεδομένων, χρησιμοποιώντας μια παραλλαγή της γνωστής στρατηγικής διαίρεσης και κατάκτησης (divide-and-conquer).

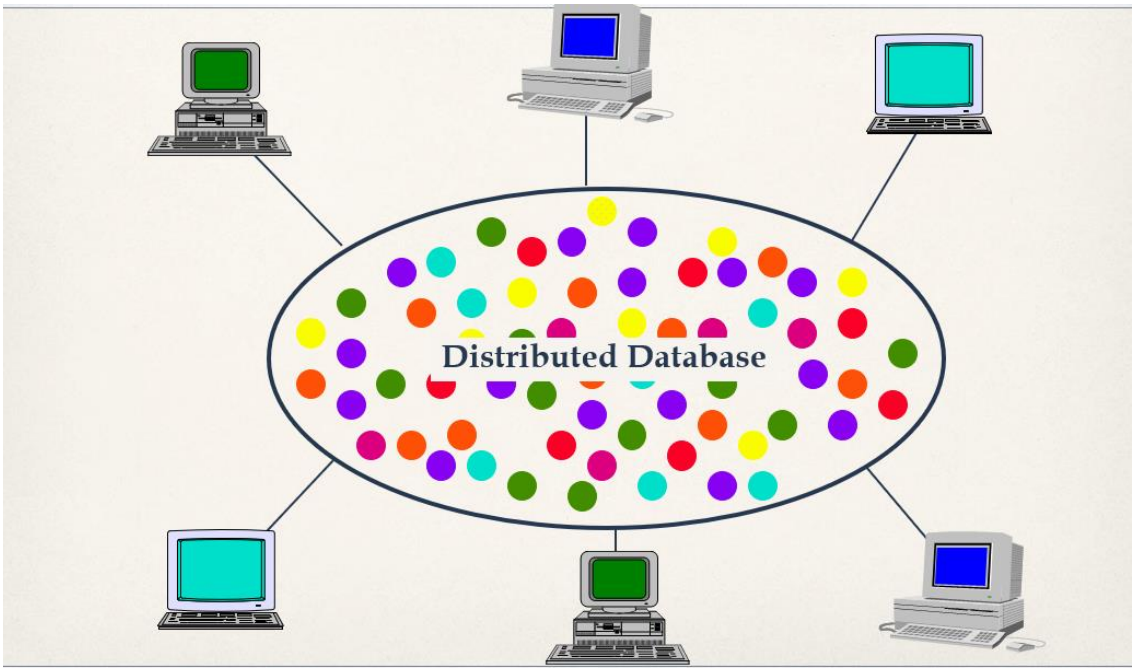
3.1 Κατανεμημένες Βάσεις Δεδομένων και Κατανεμημένα Συστήματα Διαχείρισης Βάσεων Δεδομένων (DBMS)

Ως κατανεμημένη βάση δεδομένων μπορεί να οριστεί ένα σύνολο από λογικά συνδεόμενες βάσεις δεδομένων, οι οποίες βρίσκονται σε διάσπαρτα σημεία σε ένα δίκτυο υπολογιστών (Εικόνα 14).



Εικόνα 14: Κατανεμημένες βάσεις δεδομένων λογικά συνδεδεμένες- Distributed DBMS - M.T. Özsu & P. Valduriez

Αντίστοιχα, ένα κατανεμημένο σύστημα διαχείρισης βάσεων δεδομένων περιγράφεται σαν ένα πρόγραμμα που επιτρέπει τη διαχείριση μιας κατανεμημένης βάσης δεδομένων με τέτοιο τρόπο που η κατανομή να είναι διαφανή στους χρήστες του συστήματος (Εικόνα 15) (Özsu & Valduriez, 2011).



Εικόνα 15: Κατανεμημένες βάσεις δεδομένων όπου παρουσιάζονται ως μια συνολική βάση δεδομένων - User View- Distributed DBMS - M.T. Özsu & P. Valduriez

Σε αυτό το σημείο είναι σημαντικό να τονιστεί πως μια κατανεμημένη βάση δεδομένων δεν περιγράφει ένα σύνολο τοπικών βάσεων δεδομένων που έχουν κατανεμηθεί τυχαία στο χώρο και επικοινωνούν μεταξύ τους. Προφανώς εφαρμόζεται μια λογική συνοχή των τοπικών βάσεων και στον τρόπο πρόσβασης τους χωρίς να είναι αναγκαίο ο χρήστης ή προγραμματιστής να γνωρίζει την εσωτερική δομή της κατανομής.

3.1.1 Χαρακτηριστικά Κατανεμημένων Βάσεων Δεδομένων

Κατά τη διάρκεια των τελευταίων δεκαετιών παρατηρήθηκε πως οι μεγάλοι οργανισμοί τείνουν να εκμεταλλεύονται το σύνολο των δεδομένων τους, που βρίσκεται αποθηκευμένο σε διάφορα τοπικά συστήματα μηχανογράφησης. Επομένως η ανάγκη για τον τεμαχισμό μιας βάσης δεδομένων και η γεωγραφική κατανομή της έγινε φανερή.

Η χρήση κατανεμημένων βάσεων δεδομένων προσφέρει πλεονεκτήματα όπως οργανωτικά/οικονομικά (αποδοτικότερη συντήρηση συγκριτικά με τα κεντρικά συστήματα), αποτελεσματικότερη αξιοποίηση πληροφορίας (πρόκειται για μια φυσική λύση για γεωγραφικά κατανεμημένα δεδομένα που χρησιμοποιούνται από κεντρικές εφαρμογές), ευκολότερη ανάπτυξη του μεγέθους του οργανισμού, μείωση του τηλεπικοινωνιακού φορτίου, εξασφάλιση αξιοπιστίας και διαθεσιμότητας της πληροφορίας (μέσω πολλαπλών κατανεμημένων αντιγράφων).

Όσον αφορά τα κατανεμημένα συστήματα βάσεων δεδομένων, περιγράφονται από τα παρακάτω βασικά χαρακτηριστικά:

- Έλεγχος του συστήματος. Το σύστημα ελέγχεται τοπικά από τους τοπικούς διαχειριστές αλλά και καθολικά από έναν επιβλέποντα διαχειριστή.
- Ανεξαρτησία από τα δεδομένα. Πρόκειται για ένα από τα πιο σημαντικά χαρακτηριστικά των κατανεμημένων βάσεων δεδομένων, το γεγονός πως η φυσική οργάνωση των δεδομένων είναι διαφανής στον προγραμματιστή.
- Δημιουργία αντιγράφων. Η ύπαρξη αντιγράφων είναι επιδιωκόμενη καθώς αυξάνει την απόδοση του συστήματος σε καθολικό επίπεδο.
- Βελτιστοποίηση των ερωτήσεων. Όταν παρατηρείται βελτιστοποίηση των ερωτήσεων σε περιβάλλοντα κατανεμημένων βάσεων δεδομένων, το συνολικό σύστημα βελτιώνεται καθολικά.

- Αξιοπιστία, έλεγχος συντονισμού, ανάνηψη των συστημάτων, δυνατότητα ορισμού δικαιωμάτων στους χρήστες.

3.1.2 Πιθανές πολυπλοκότητες κατά τη χρήση κατανομής δεδομένων

Η επιλογή της σωστής βάσης δεδομένων αποτελεί μια σημαντική διαδικασία κατά την ανάπτυξη μιας εφαρμογής καθώς μπορεί να επιφέρει διάφορες συνέπειες στο βραχυπρόθεσμο μέλλον. Η αναγνώριση μη-συμβατότητας είναι ένα χαρακτηριστικό που είναι προτιμότερο να εντοπιστεί στα αρχικά στάδια, καθώς είναι ευκολότερο το σύστημα να μεταφέρει τις ανάγκες του σε μια άλλη βάση δεδομένων που θα καλύπτει αποδοτικότερα τους στόχους της εφαρμογής.

Χωρίς αμφιβολία πολλά από τα προβλήματα που είναι δυνατόν να δημιουργηθούν σε ένα σύστημα βάσης δεδομένων εμφανίζονται με έναν επιπλέον βαθμό πολυπλοκότητας στις περιπτώσεις των κατανεμημένων βάσεων δεδομένων.

Αρχικά, είναι πολύ πιθανό πως τα δεδομένα θα αναπαραχθούν σε ένα κατανεμημένο περιβάλλον. Για την ακρίβεια οι περισσότερες βάσεις δεδομένων σχεδιάζονται με τέτοιο τρόπο όπου το κάθε τμήμα μια βάσης δεδομένων βρίσκεται σε διαφορετικές τοποθεσίες ενός δικτύου υπολογιστών. Δεν είναι απαραίτητο πως ο κάθε κόμβος του δικτύου θα κατέχει κάποιο μέρος της βάσης δεδομένων, αρκεί η βάση δεδομένων να είναι κατανεμημένη σε τουλάχιστον δύο τοποθεσίες του δικτύου.

Όπως γίνεται φανερό λοιπόν η πιθανή επανάληψη δεδομένων γίνεται πιθανή και οφείλεται κυρίως σε λόγους αξιοπιστίας και αποτελεσματικότητας. Επομένως, το κατανεμημένο σύστημα είναι υπεύθυνο για την επιλογή ενός από τα διαθέσιμα αντίγραφα του ζητούμενου δεδομένου αλλά και παράλληλα για την ενημέρωση όλων των αντιγράφων για πιθανές αλλαγές και ενημερώσεις.

3.2 Ομογενείς και Ετερογενείς Κατανεμημένες βάσεις δεδομένων

Μια κατανεμημένη βάση δεδομένων μπορεί να κατηγοριοποιηθεί σε δύο χαρακτηριστικά είδη με βάση τον τύπο ή πολλαπλούς διάφορους τύπους κατανεμημένων συστημάτων που χρησιμοποιεί ο κάθε κόμβος του κατανεμημένου συστήματος. Συγκεκριμένα όταν ένα σύστημα ορίζεται ως ομογενή όλα τα μέρη/κόμβοι του συστήματος χρησιμοποιούν το ίδιο

προϊόν συστήματος διαχείρισης βάσεων δεδομένων και τα δεδομένα είναι διαμοιρασμένα μεταξύ των κόμβων. Σε αντίθεση, σε ένα ετερογενές σύστημα δύναται οι κόμβοι να εκτελούν διαφορετικούς τύπους συστήματος διαχείρισης βάσεων δεδομένων, τα οποία μπορεί να διαφέρουν ριζικά μεταξύ τους και να ανήκουν σε διαφορετικές κατηγορίες συστήματος διαχείρισης βάσης δεδομένων (Σχεσιακά, δικτυακά, ιεραρχικά ή αντικειμενοστραφή ΣΔΒΔ) (Thakur, n.d.).

Έχει παρατηρηθεί πως τα ομογενή συστήματα είναι ουσιαστικά ευκολότερα όσον αφορά το σχεδιασμό και τη διαχείριση τους σε σχέση με τα ετερογενή. Η παραπάνω προσέγγιση επιτρέπει τη σταδιακή ανάπτυξη του συστήματος, καθώς η προσθήκη ενός νέου κόμβου στο σύστημα καθίσταται απλούστερη και μπορεί να εκμεταλλευτεί τις δυνατότητες παράλληλης επεξεργασίας με σκοπό τη βελτίωση της απόδοσης.

Από την άλλη πλευρά, πολύ συχνά οι μεμονωμένοι κόμβοι εφαρμόζουν το δικό τους σχεδιασμό της βάσης δεδομένων και οι ενοποίηση του με άλλα συστήματα ή κόμβους διερευνάται σε μεταγενέστερο στάδιο, με αποτέλεσμα να σχηματίζεται ένα ετερογενές σύστημα.

Για την ομαλή επικοινωνία μεταξύ ετερογενών συστημάτων απαιτούνται οι ανάλογες μεταφράσεις. Επίσης για λόγους διαφάνειας στα ετερογενή συστήματα, οι χρήστες θα πρέπει να υποβάλουν αιτήματα χρησιμοποιώντας την ανάλογη γλώσσα ΣΔΒΔ που χρησιμοποιεί ο τοπικός κόμβος του. Στη συνέχεια, το σύστημα είναι υπεύθυνο για τον εντοπισμό των δεδομένων και για την πραγματοποίηση των απαραίτητων μεταφράσεων. Τα δεδομένα που ζητούνται ενδέχεται να προέρχονται από έναν κόμβο όπου χρησιμοποιεί διαφορετικό εξοπλισμό, διαφορετικό τύπο ΣΔΒΔ ή ένα συνδυασμό διαφορετικού εξοπλισμού και τύπου ΣΔΒΔ.

Όλα αυτά έχουν ως αποτέλεσμα ορισμένα συστήματα που αποτελούν μέρος ενός ετερογενούς συστήματος να χρησιμοποιούν πύλες (gateways) για τη μετατροπή της γλώσσας και του μοντέλου του κάθε συστήματος διαχείρισης βάσης δεδομένων στη γλώσσα και τον τύπο του σχεσιακού συστήματος.

3.3 Τεχνικές Κατανομής και Αποθήκευσης Δεδομένων κατά το σχεδιασμό κατανεμημένων βάσεων δεδομένων

Κατά το στάδιο του σχεδιασμού ενός συστήματος που έχει αποφασιστεί πως θα υιοθετήσει την τεχνική της κατανεμημένης διαχείρισης δεδομένων, ένα ιδιαίτερα σημαντικό κομμάτι της διαδικασίας είναι πως ο προγραμματιστής της βάσης δεδομένων γνωρίζει και κατανοεί όλες τις διαθέσιμες τεχνικές που είναι δυνατό να χρησιμοποιούν. Για το σκοπό αυτό, παρακάτω γίνεται αναφορά σε κάποιες από τις βασικές τεχνικές κατανομής και αποθήκευσης δεδομένων που συνηθίζεται να επιλέγονται κατά το σχεδιασμός ενός συστήματος με χρήση κατανεμημένων βάσεων δεδομένων.

3.3.1 Τμηματοποίηση (Fragmentation)

Το έργο του κατακερματισμού ενός συνόλου δεδομένων σε μικρότερα τμήματα δεδομένων είναι γνωστό ως Τμηματοποίηση ή κατακερματισμός δεδομένων.

Υπάρχουν τρία βασικά είδη τμηματοποίησης: οριζόντιος, κάθετος και υβριδικός (συνδυασμός κάθετου και οριζόντιου), για τα οποία θα γίνει αναλυτικότερη αναφορά παρακάτω.

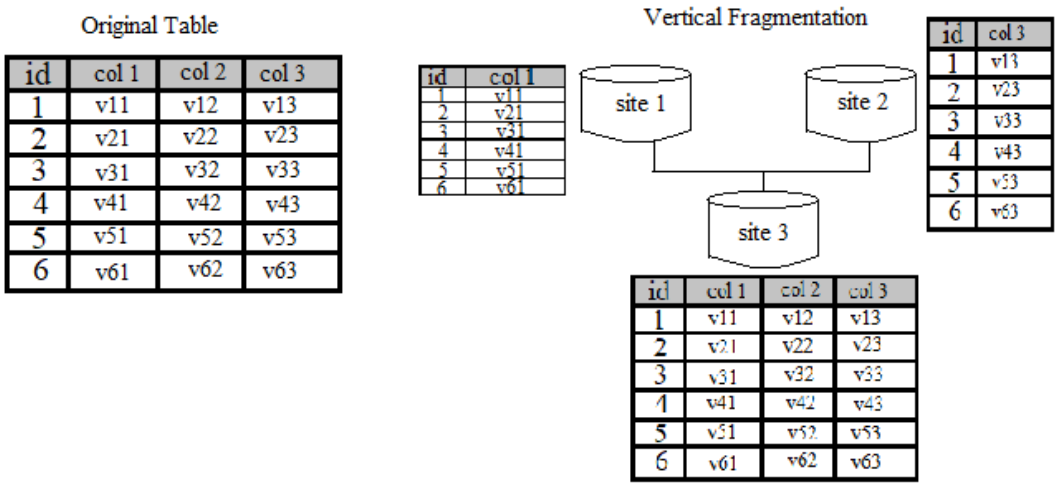
Εξαιρετική σπουδαιότητα κατά τη διαδικασία της Τμηματοποίησης κατέχει το χαρακτηριστικό της ανακατασκευής του αρχικού συνόλου. Τα τμήματα δεδομένων είναι απαραίτητο να έχουν κατανεμηθεί με τέτοιο τρόπο όπου ο συνδυασμός τους μπορεί να ανακατασκευάσει το αρχικό σύνολο πληροφοριών σε περίπτωση που χρειαστεί.

Αξίζει, επιπλέον, να γίνει αναφορά σε μερικά από τα πλεονεκτήματα της χρήσης της Τμηματοποίησης. Αρχικά, η αποτελεσματικότητα του συστήματος βάσεων δεδομένων βελτιώνεται δεδομένου ότι τα δεδομένα αποθηκεύονται κοντά στο σημείο χρήσης. Ακόμα, διότι τα δεδομένα είναι διαθέσιμα τοπικά, οι τοπικές τεχνικές βελτιστοποίησης ερωτημάτων επαρκούν για τα περισσότερα ερωτήματα. Τέλος, δεδομένου ότι δεν υπάρχουν διαθέσιμα μη-σχετικά δεδομένα στους κόμβους, είναι δυνατόν να διατηρηθεί η ασφάλεια και το απόρρητο του συστήματος βάσεων δεδομένων

Αντίστοιχα, η τεχνική της τμηματοποίησης φέρει κάποια μειονεκτήματα, τα οποία είναι χρήσιμο να ληφθούν υπ’ όψιν, όπως τις περιπτώσεις όπου απαιτούνται δεδομένα από πολλαπλά τμήματα, οι χρόνοι πρόσβασης μπορεί να είναι εξαιρετικά γρήγοροι. Επιπλέον, στην περίπτωση αναδρομικών κατακερματισμών, η ανοικοδόμηση θα απαιτήσει τη χρήση

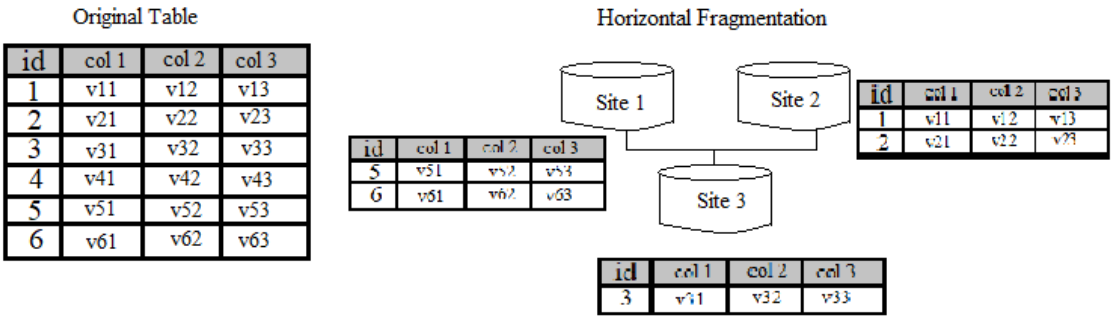
δαπανηρών διαδικασιών. Χρειάζεται επίσης, να σημειωθεί ότι η έλλειψη αντιγράφων ασφάλειας δεδομένων σε πολλαπλές τοποθεσίες μπορεί να καταστήσει τη βάση δεδομένων αναποτελεσματική σε περίπτωση αποτυχίας ενός κόμβου.

Ο κατακόρυφος/κάθετος κατακερματισμός ομαδοποιεί τα πεδία ή τις στήλες ενός πίνακα δεδομένων σε τμήματα Κάθε τμήμα οφείλει να έχει το πρωταρχικό πεδίο κλειδιού του πίνακα για τη διατήρηση της ανοικοδόμησης. Το σχήμα της Εικόνας 16 επεξηγεί τον τρόπο ομαδοποίησης κατά τον κάθετο κατακερματισμό χρησιμοποιώντας ένα παράδειγμα μιας τυπικής βάσης δεδομένων. Συγχρόνως έχει παρατηρηθεί πως το απόρρητο των δεδομένων μπορεί να ενισχυθεί με κατακόρυφο κατακερματισμό.



Εικόνα 16: Κατακόρυφος κατακερματισμός

Κατά τον οριζόντιο κατακερματισμό ομαδοποιούνται οι τις πλειάδες ενός πίνακα σε ομάδες με βάση τις τιμές ενός ή περισσοτέρων πεδίων. Ο κανόνας της ανασυγκρότησης πρέπει επίσης να ισχύει για τον οριζόντιο κατακερματισμό. Όλες οι στήλες από τον αρχικό πίνακα βάσης πρέπει να υπάρχουν σε κάθε οριζόντιο τμήμα, κάτι το οποίο γίνεται φανερό στην Εικόνα 17.



Εικόνα 17: Οριζόντιος Κατακερματισμός

Σε συνάρτηση με τα προηγούμενα είδη κατακερματισμού, ο συνδυασμός τεχνικών οριζόντιου και κατακόρυφου κατακερματισμού χρησιμοποιείται στον υβριδικό κατακερματισμό. Επειδή δημιουργεί τμήματα με τη μικρότερη ποσότητα εξωτερικών πληροφοριών, αυτή είναι η πιο προσαρμόσιμη τεχνική κατακερματισμού. Ωστόσο, η αναδημιουργία του αρχικού πίνακα είναι συχνά μια δαπανηρή διαδικασία (Shah, 2021).

3.3.2 Αντιγραφή και Κατανομή (Replication and Allocation)

Ως αντιγραφή δεδομένων ορίζεται η διαδικασία αποθήκευσης ξεχωριστών αντιγράφων της βάσης δεδομένων σε δύο ή περισσότερους κόμβους. Αποτελεί μια δημοφιλή τεχνική ανοχής σφαλμάτων κατανεμημένων βάσεων δεδομένων. Η πιο ακραία μορφή αυτής της τεχνικής είναι η πλήρης αναπαραγωγή βάσης δεδομένων σε όλους τους κόμβους ενός κατανεμημένου συστήματος, με αποτέλεσμα μια πλήρως αντιγραφόμενη κατανεμημένη βάση δεδομένων.

Επειδή ακριβώς το σύστημα μπορεί να συνεχίσει να λειτουργεί όσο τουλάχιστον ένας κόμβος βρίσκεται σε λειτουργία, μπορεί να βελτιώσει δραματικά τη διαθεσιμότητα. Βελτιώνει επίσης την απόδοση ανάκτησης για καθολικά αιτήματα επιτρέποντας τα αποτελέσματα τέτοιων ερωτημάτων να εξυπηρετούνται τοπικά από οποιαδήποτε τοποθεσία. Ως εκ τούτου, ένα αίτημα μπορεί να χειριστεί από τον τοπικό κόμβο όπου λαμβάνει το αίτημα εάν αυτός ο συγκεκριμένος τοπικός κόμβος διαθέτει μονάδα διακομιστή.

Η πλήρης αντιγραφή έχει το μειονέκτημα της επιβράδυνσης των λειτουργιών ενημέρωσης, διότι πρέπει να εκτελείται μία λογική ενημέρωση σε κάθε αντίγραφο της βάσης δεδομένων, ώστε τα αντίγραφα να είναι συνεπή. Αυτό ισχύει ιδιαίτερα εάν υπάρχουν πολλά αντίγραφα της βάσης δεδομένων. Οι προσεγγίσεις ελέγχου ταυτότητας και ανάκτησης είναι πιο ακριβές με πλήρη αναπαραγωγή από ότι θα ήταν χωρίς αυτό.

Το αντίθετο της πλήρους αντιγραφής δεν είναι η μηδενική αντιγραφή, κατά την οποία το κάθε τμήμα δεδομένων αποθηκεύεται σε μία ακριβώς τοποθεσία. Εκτός από την επανάληψη πρωτευόντων κλειδιών μεταξύ κάθετων (ή μικτών) κομματιών, όλα τα τμήματα πρέπει να βρίσκονται σε ξεχωριστούς κόμβους σε αυτό το σενάριο. Η αχρησιμοποίητη κατανομή (nonredundant allocation) είναι ένας άλλος όρος για αυτή την προσέγγιση.

Κάθε τμήμα (ή αντίγραφο ενός τμήματος) στο κατανεμημένο σύστημα πρέπει να αντιστοιχιστεί σε έναν συγκεκριμένο κόμβο. Η κατανομή δεδομένων είναι ο όρος για αυτήν τη διαδικασία (ή καταμερισμός δεδομένων). Ο αριθμός των κόμβων και ο βαθμός αντιγραφής καθορίζονται από τους στόχους απόδοσης και διαθεσιμότητας του συστήματος, καθώς και από τους τύπους και τη συχνότητα των συναλλαγών που υποβάλλονται σε κάθε τοποθεσία. Η εύρεση μιας βέλτιστης ή μιας αρκετά καλής λύσης κατανομής δεδομένων αποτελεί ένα δύσκολο πρόβλημα βελτιστοποίησης

(Wiese, 2014).

4. Εφαρμογή Κατανεμημένης Διαχείρισης Προσωπικών Δεδομένων

Ο σκοπός του παρόν κεφαλαίου είναι η παρουσίαση της σχετικής κατανεμημένης εφαρμογής που αναπτύχθηκε στα πλαίσια της παρούσας εργασίας και των σχετικών τεχνολογιών και εργαλείων που χρησιμοποιήθηκαν.

4.1 Στόχοι της εφαρμογής

Κατά την έρευνα του θέματος της παρούσας πτυχιακής εργασίας γίνεται φανερή η ανάγκη συστημάτων τα οποία εστιάζουν στην κατανεμημένη αποθήκευση δεδομένων, την ιδιοκτησία αυτών αλλά και την κοινή χρήση των πληροφοριών με άλλους χρήστες υπό ορισμένες προϋποθέσεις. Με βάση την παραπάνω ανάγκη αναπτύχθηκε μια εφαρμογή με βασικό στόχο την κατανεμημένη αποθήκευση των δεδομένων του χρήστη σε διάφορους κόμβους και την κοινή χρήση τους μέσω της συνεργασίας πολλαπλών κόμβων και κατανεμημένη διαχείριση των δεδομένων με σκοπό την άμεση και αποδοτικότερη εξυπηρέτηση.

Τα τελευταία χρόνια, λαμβάνοντας υπ' όψιν τη ραγδαία ανάπτυξη του υλικού και του λογισμικού στο χώρο της τεχνολογίας, ο όγκος των δεδομένων που καταγράφονται και χρησιμοποιούνται, τόσο σε εμπορικό όσο και σε επιστημονικό επίπεδο, είναι σημαντικά μεγαλύτερος. Επομένως είναι αναμενόμενο πως απαιτούνται νέες αποδοτικότερες τεχνικές διαχείρισης όταν πρόκειται για ένα σημαντικό όγκο πληροφοριών.

Κατά συνέπεια, ένα σοβαρό ζήτημα της μαζικής καταγραφής δεδομένων από τους χρήστες των σύγχρονων εφαρμογών είναι η ιδιοκτησία, η κοινή χρήση και η γενικότερη προστασία των δεδομένων. Το κάθε σύστημα είναι υπεύθυνο να προστατεύσει τα δεδομένα των χρηστών που διαχειρίζεται και να αποτρέπει μη-εξουσιοδοτημένες ενέργειες. Ο χρήστης θα πρέπει να έχει τη δυνατότητα να επιλέξει ποιος έχει πρόσβαση στα δεδομένα του και να έχει πλήρη κατανόηση της πολιτικής σχετικά με την ιδιοκτησία των δεδομένων του.

Το ζήτημα της ιδιοκτησίας των δεδομένων έχει κεντρίσει το ενδιαφέρον των μελετητών τα τελευταία χρόνια. Ως δεδομένα ορίζονται οι πληροφορίες που παρέχονται από τους χρήστες και λαμβάνοντας υπ' όψιν το πλαίσιο και τη χρήση των δεδομένων αποκτούν την ανάλογη αξία. Ενώ υπάρχουν διάφοροι ορισμοί του όρου «ιδιοκτησία δεδομένων», οι

οποίοι διαφέρουν ανάλογα το πλαίσιο, τη φύση των δεδομένων, τους νόμους που εφαρμόζονται, κ.α., για την ανάπτυξη της σχετικής εφαρμογής ως ιδιοκτησία δεδομένων θεωρείται η δυνατότητα του χρήστη να εισάγει και να αφαιρέσει νέα δεδομένα στο σύστημα, τη δυνατότητα επιλογής κοινής χρήσης των δεδομένων με άλλους χρήστες αλλά και την απόσυρση αυτής οποιαδήποτε στιγμή. Ένας χρήστης του συστήματος δεν έχει πρόσβαση σε δεδομένα που δεν του έχει επιτραπεί η πρόσβαση από τον ιδιοκτήτη του εγγράφου. Έχει τη δυνατότητα να αποστείλει μια σχετική αίτηση πρόσβασης για κάποιο δεδομένο και εφόσον του επιτραπεί η πρόσβαση αποκτά τα σχετικά δικαιώματα. Παράλληλα ο χρήστης έχει την ελευθέρια να αφαιρέσει το δικαίωμα πρόσβασης κάποιου χρήστη στα δεδομένα του, ακόμη και αν έχει αποδεχτεί το αίτημα πρόσβασης στο παρελθόν.

Ο στόχος της εφαρμογής που αναπτύχθηκε είναι η κατανεμημένη διαχείριση και διασφάλιση προσωπικών δεδομένων των χρηστών της. Καθώς πρόκειται για μια κατανεμημένη εφαρμογή, οι χρήστες μπορούν να εξυπηρετούνται από διαφορετικούς κόμβους με σκοπό να αποφευχθούν πιθανές καθυστερήσεις που συμβαίνουν σε μη-κατανεμημένα συστήματα λόγω υψηλού φόρτου εργασίας. Τα δεδομένα των χρηστών αποθηκεύονται τοπικά στον ανάλογο κόμβο που διαχειρίζεται τον κάθε χρήστη και περιοδικά πραγματοποιείται η μακροπρόθεσμη αντιγραφή των δεδομένων στη βάση δεδομένων. Ακόμα, επιτρέπεται η κοινή χρήση δεδομένων, μετά από το ανάλογο αίτημα πρόσβασης, μεταξύ των χρηστών.

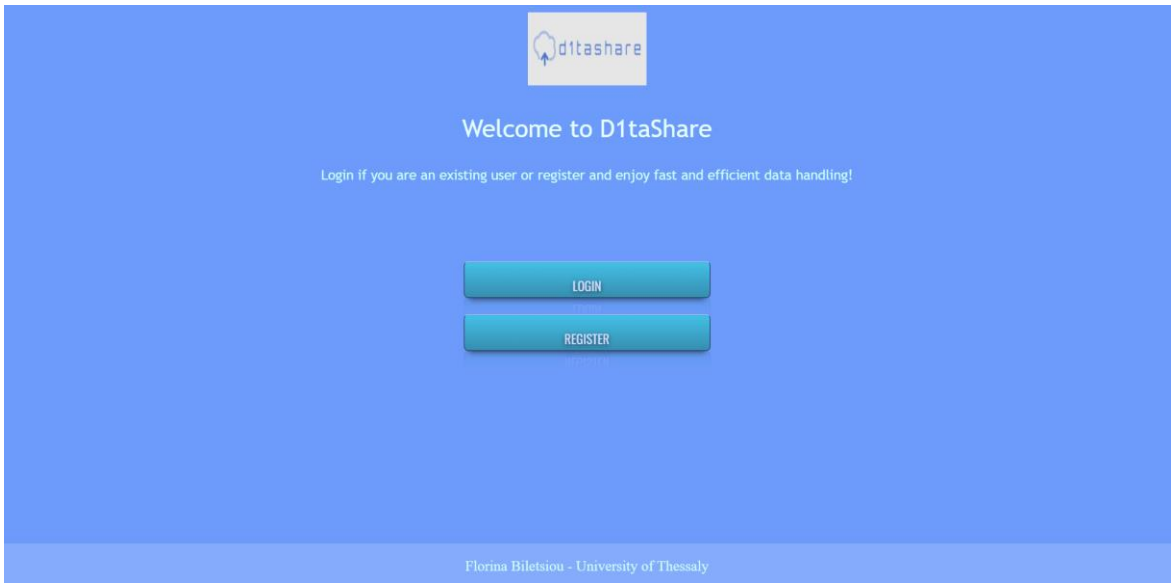
4.2 Περιγραφή εφαρμογής

4.2.1 Σύντομη περιήγηση εφαρμογής και εμπειρίας του χρήστη

Παρακάτω ακολουθεί μια σύντομη και τυπική περιήγηση της πορείας του χρήστη στο σύστημα.

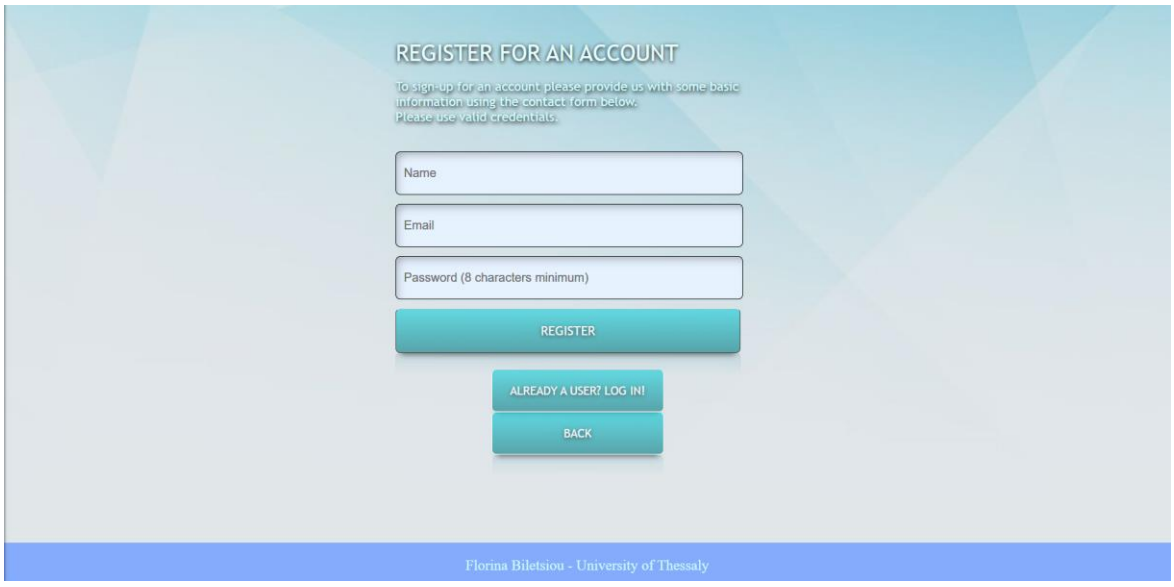
Ο χρήστης αρχικά εγγράφεται στο σύστημα ώστε να δημιουργηθεί ο λογαριασμός του και να αποκτήσει τη δυνατότητα να αποθηκεύει τα προσωπικά του δεδομένα στο σύστημα.

Αρχικά, όπως εμφανίζεται στην Εικόνα 18, οι χρήστες καλωσορίζονται από την αρχική σελίδα που τους επιτρέπει είτε να εγγραφούν στο σύστημα είτε να συνδεθούν στο λογαριασμό τους.



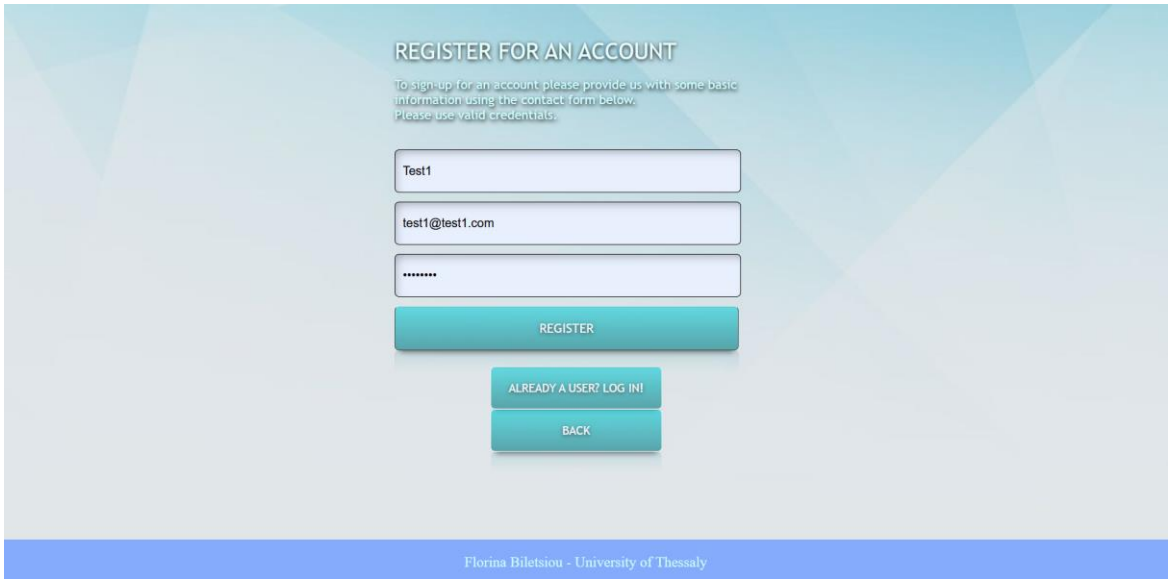
Εικόνα 18: Αρχική σελίδα της εφαρμογής – Ο χρήστης έχει την επιλογή δημιουργίας νέου λογαριασμού ή σύνδεσης στον ήδη υπάρχοντα.

Έστω πρόκειται για νέο χρήστη, οδηγείται στη φόρμα εγγραφής όπου πρέπει να συμπληρώσει τα στοιχεία του και τον επιθυμητό κωδικό πρόσβασης (Εικόνα 19).



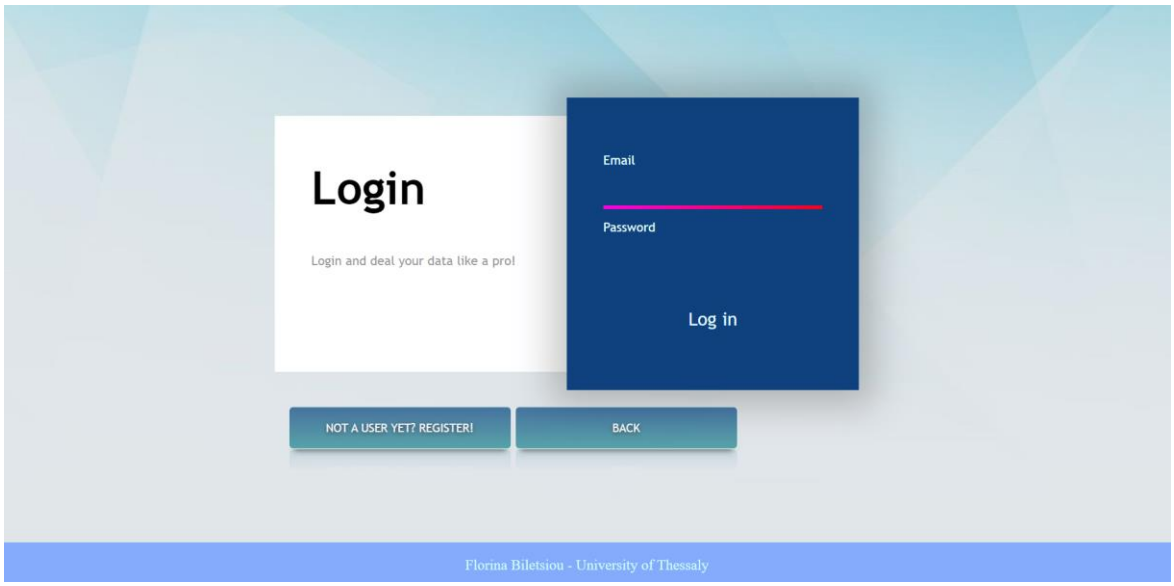
Εικόνα 19: Φόρμα εγγραφής νέου χρήστη.

Ο χρήστης αφού συμπληρώσει τα απαιτούμενα στοιχεία του για τη διαδικασία της εγγραφής, αποστέλλει το αίτημα στο σύστημα επιλέγοντας το κουμπί «REGISTER», όπως φαίνεται στην Εικόνα 20.



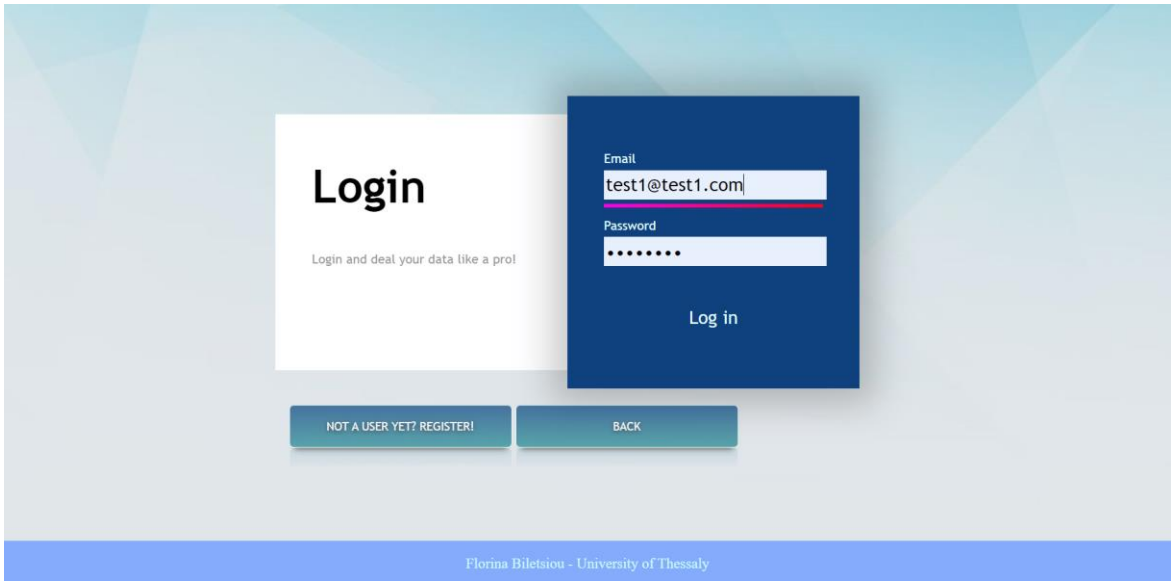
Εικόνα 10: Ο χρήστης αποστέλλει το αίτημα εγγραφής αφού συμπληρώσει τα απαιτούμενα στοιχεία του.

Αμέσως μετά την ολοκλήρωση της εγγραφής του νέου χρήστη, ανακατευθύνεται στη σελίδα σύνδεσης (Εικόνα 21), όπου πρέπει να συμπληρώσει το email και τον κωδικό που επέλεξε στο προηγούμενο βήμα. Επίσης ο χρήστης ανακατευθύνεται στη συγκεκριμένη σελίδα σύνδεσης εάν επιλέξει την επιλογή «login» της αρχικής σελίδας της Εικόνας 18.



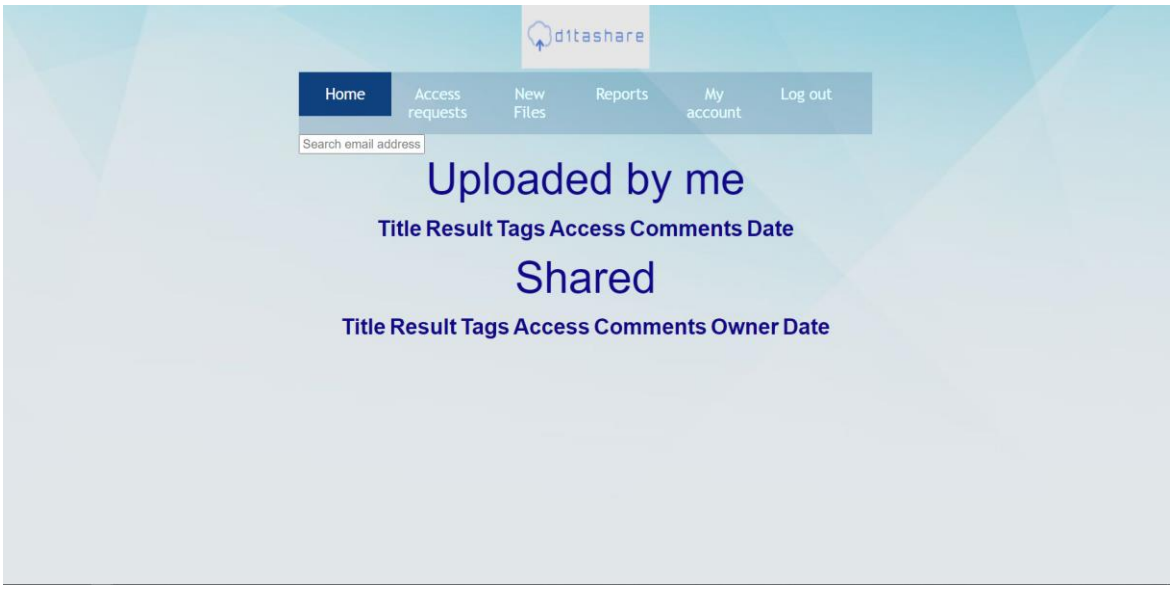
Εικόνα 11: Σελίδα σύνδεσης χρήστη.

Όταν ο χρήστης συμπληρώσει τη διεύθυνση ηλεκτρονικού ταχυδρομείου του και τον επιλεγμένο κωδικό του το επόμενο βήμα για τη σύνδεση του στο σύστημα είναι η χρήση της ενέργειας «Log in» της Εικόνας 22.



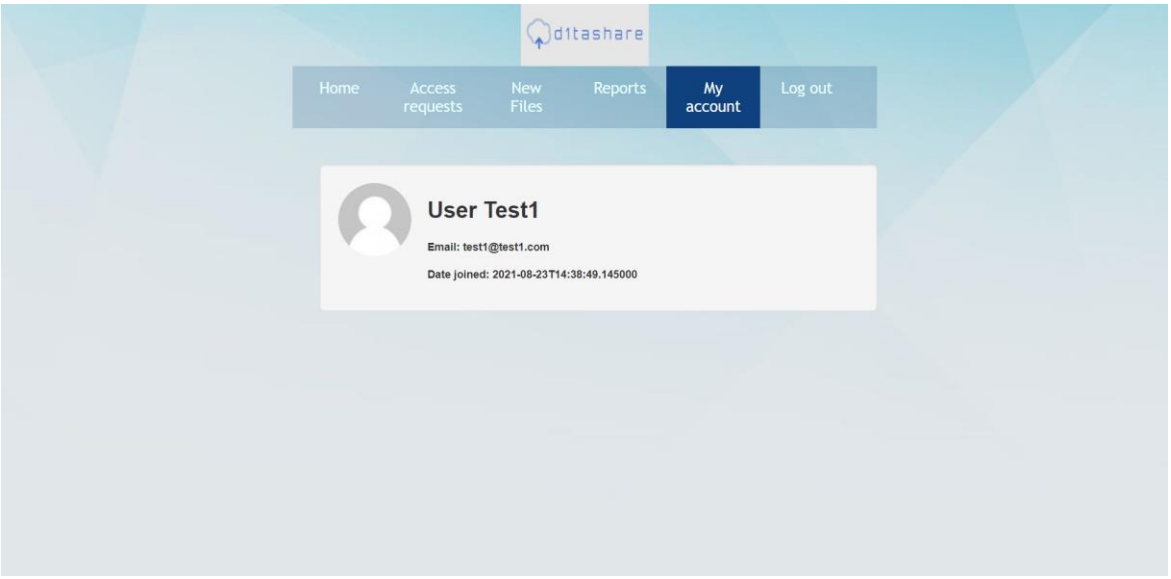
Εικόνα 12: Ο χρήστης συμπληρώνει τα στοιχεία του και επιλέγει το κουμπί «Log in» για να συνδεθεί στο σύστημα.

Εφόσον ο χρήστης συνδεθεί επιτυχώς έχει πλέον πρόσβαση στην κύρια σελίδα της εφαρμογής που εμφανίζονται τα δεδομένα που έχει εισάγει στο σύστημα και τα δεδομένα που έχουν κοινοποιεί μετά από το ανάλογο αίτημα πρόσβασης. Στην Εικόνα 23 εμφανίζεται η κύρια σελίδα στο στάδιο που ο χρήστης δεν έχει ακόμα δεδομένα.



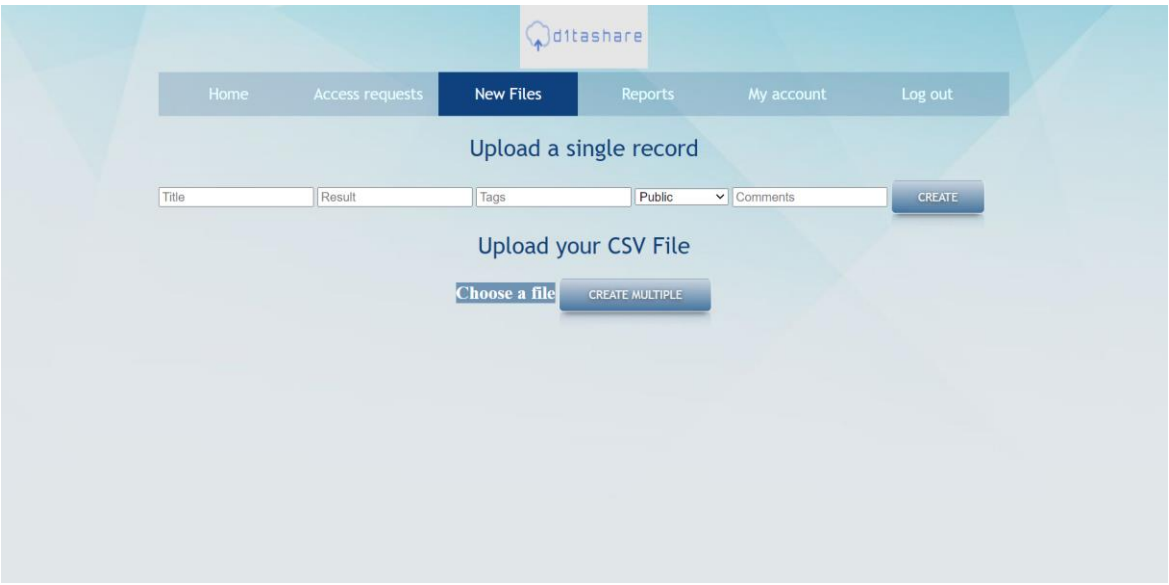
Εικόνα 13: Κύρια σελίδα χρήστη αμέσως μετά την εγγραφή του στο σύστημα

Ο χρήστης έχει τη δυνατότητα να ελέγξει τις προσωπικές πληροφορίες σχετικά με το λογαριασμό τους μέσω της επιλογής «My account», όπως φαίνεται στην Εικόνα 24.



Εικόνα 14: Σελίδα σχετική με το λογαριασμό του χρήστη στο σύστημα.

Επίσης, ο χρήστης έχει τη δυνατότητα να εισάγει νέα δεδομένα μεμονωμένα ή ομαδικά μέσω ενός csv αρχείου (Εικόνα 25).

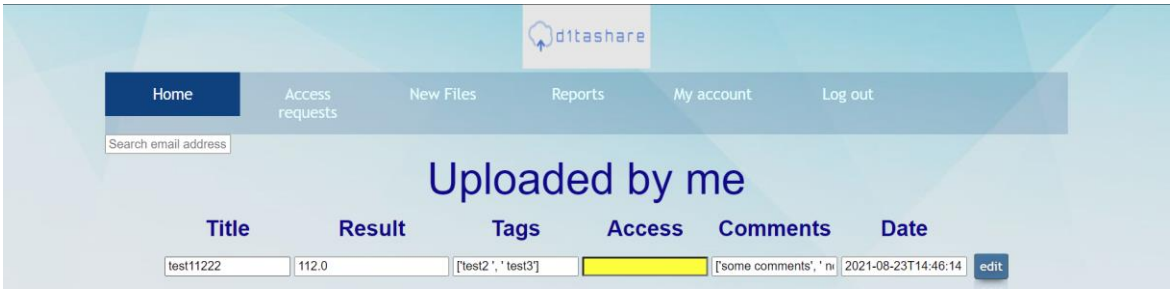


Εικόνα 15: Σελίδα εισαγωγής νέων δεδομένων από το χρήστη (μεμονωμένα ή ομαδικά).

Στην περίπτωση που ο χρήστης επιθυμεί να εισάγει ένα νέο έγγραφο μεμονωμένα το μόνο που έχει να κάνει είναι να εισάγει τις ανάλογες πληροφορίες στη φόρμα της εικόνας 26 και να πατήσει το κουμπί «Create».

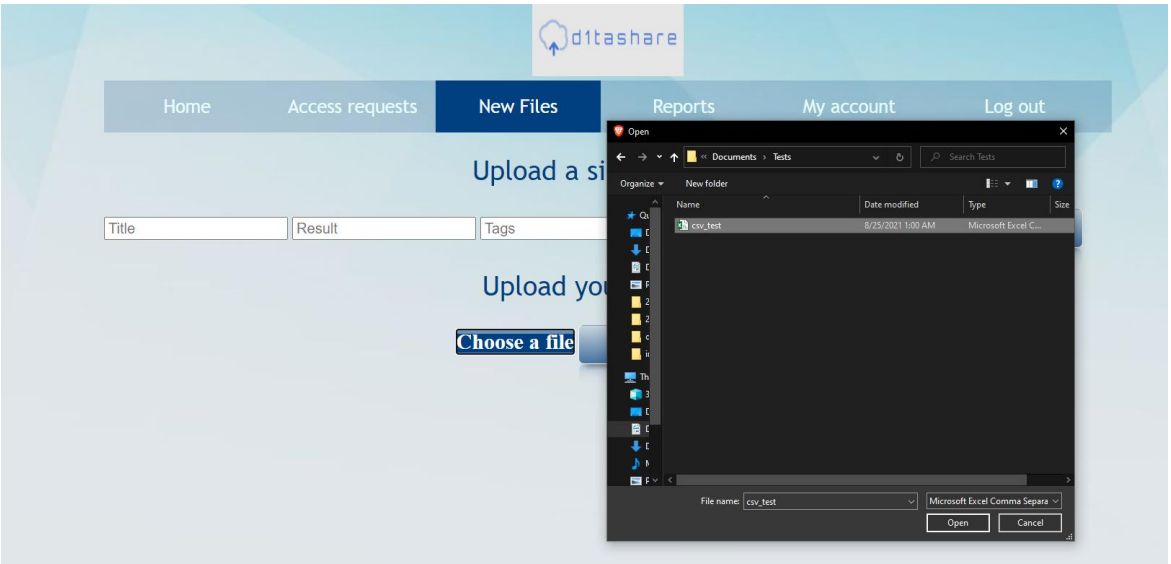
Εικόνα 16: Δημιουργία ενός μεμονωμένου εγγράφου μέσω της σελίδας εισαγωγής νέων δεδομένων.

Αμέσως μετά τη δημιουργία των νέων δεδομένων, τα νέα δεδομένα παρατίθενται στην κύρια σελίδα του χρήστη (Εικόνα 27).



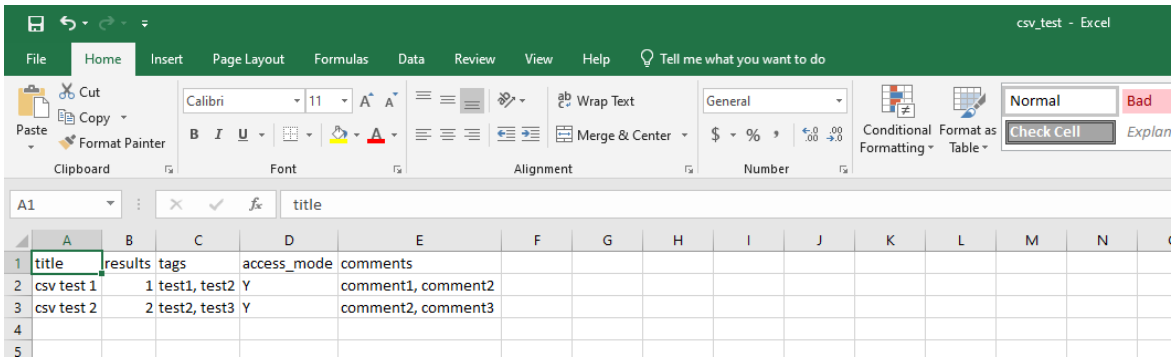
Εικόνα 17: Κύρια σελίδα χρήστη μετά την εισαγωγή νέων δεδομένων από τον ίδιο το χρήστη.

Αντίστοιχα, στην περίπτωση που ο χρήστης επιλέξει να εισάγει νέα δεδομένα μέσω ενός αρχείου τύπου csv πρέπει να κατευθυνθεί στην επιλογή «Choose a file» (Εικόνα 25). Αμέσως μετά ο χρήστης επιλέγει το ανάλογο csv αρχείο από το παράθυρο επιλογής όπου εμφανίζεται (Εικόνα 28).



Εικόνα 18: Επιλογή csv αρχείου για δημιουργία πολλαπλών δεδομένων από το χρήστη

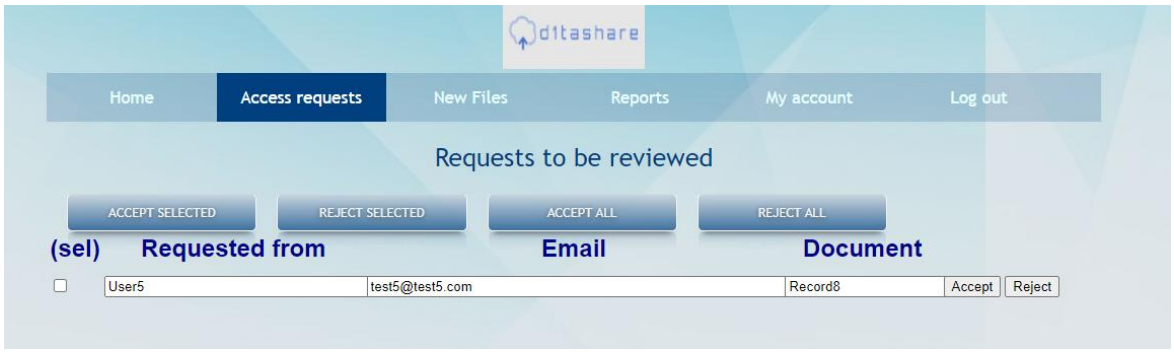
Ο χρήστης οφείλει να διασφαλίσει πως το αρχείο csv διατηρεί τη μορφοποίηση η οποία είναι συμβατή με την ανάλογη μορφή δεδομένων του συστήματος. Η παραπάνω μορφή δεδομένων παρουσιάζεται μέσω του παραδείγματος της Εικόνας 29.



Εικόνα 19: Συμβατή μορφή δεδομένων τύπου αρχείων csv.

Πιο συγκεκριμένα, το αρχείο τύπου csv πρέπει να ξεκινάει με τα ονόματα των μεταβλητών στην πρώτη γραμμή. Οι μεταβλητές που είναι απαραίτητες είναι οι εξής: title, results, access_mode. Αντίστοιχα οι μεταβλητές tags και comments είναι προαιρετικές καθώς πρόκειται για μεταβλητές που ως στόχο έχουν την περιγραφή του κάθε δεδομένου. Πάραυτα, εφόσον πρόκειται για πεδία που μπορεί να εμπεριέχουν πολλαπλά στοιχεία εντός (πίνακες δεδομένων), τα πολλαπλά στοιχεία ωφελούν να διαχωρίζονται αυστηρά με το χαρακτήρα ¨,¨ (κόμμα).

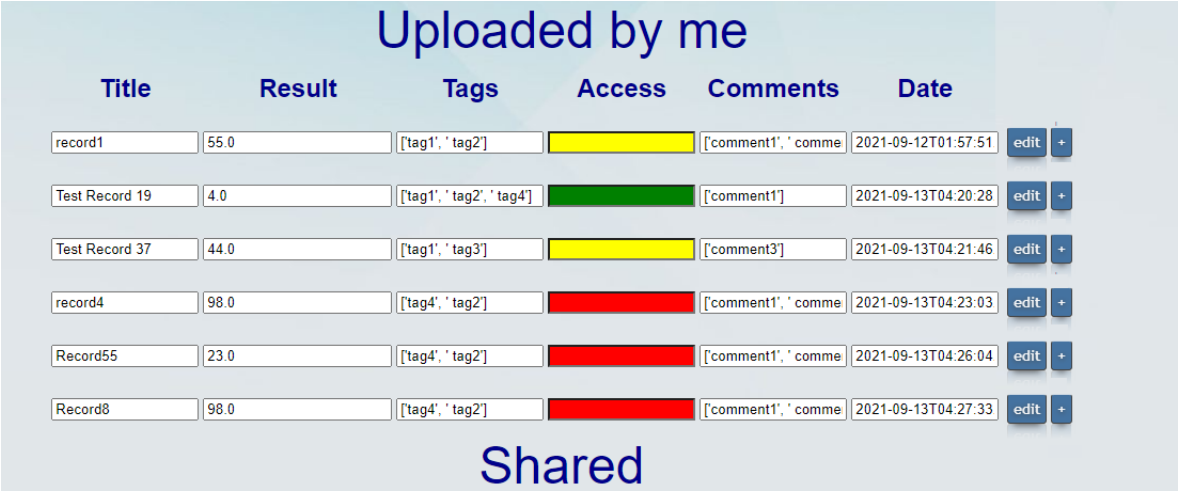
Ακόμα, ο χρήστης δέχεται αιτήματα πρόσβασης από άλλους χρήστες σχετικά με αρχεία τα οποία έχουν την επιτρεπτή ορατοτητα (Y: Accessible after request). Όπως φαινεται στην εικόνα 30, υπάρχει η δυνατότητα το αίτημα να γίνει είτε αποδεκτό είτε να απορριφθεί με τις ανάλογες ενέργειες, να επιτραπεί η πρόσβαση ή να απορριφθεί το αίτημα αντίστοιχα.



Εικόνα 30: Αίτημα πρόσβασης αρχείου και οι ανάλογες δυνατές ενέργειες, Αποδοχή και Απόρριψη αιτήματος.

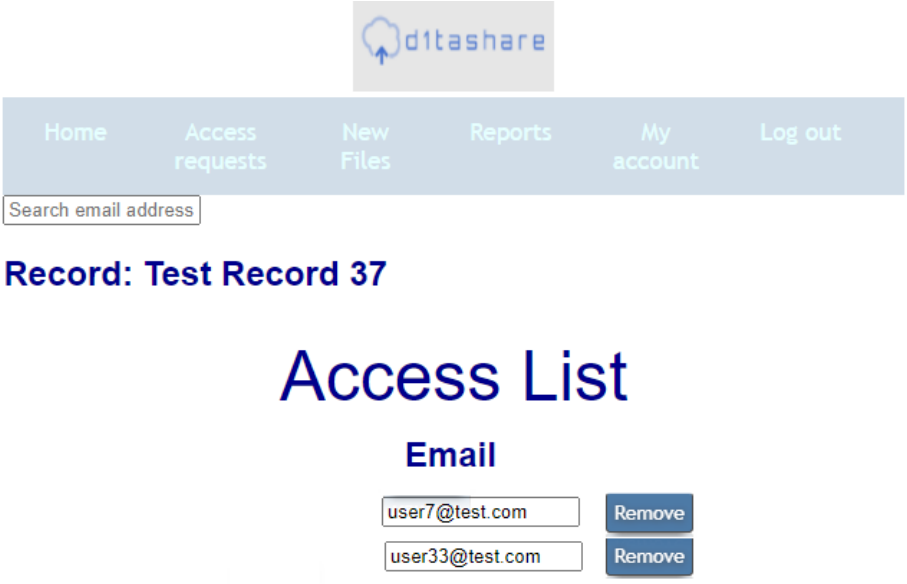
Τέλος, μια από τις σημαντικότερες πλευρές της εφαρμογής αποτελεί η διαχείριση της ιδιοκτησίας κάθε δεδομένου και η κοινή χρήση αυτού με άλλους χρήστες. Στην Εικόνα 31

ο χρήστης έχει την δυνατότητα να διαχειριστεί τα εξωτερικά δικαιώματα πρόσβασης του κάθε αρχείου, μέσω της επιλογής «+» στο δεξί μέρος της σελίδας.



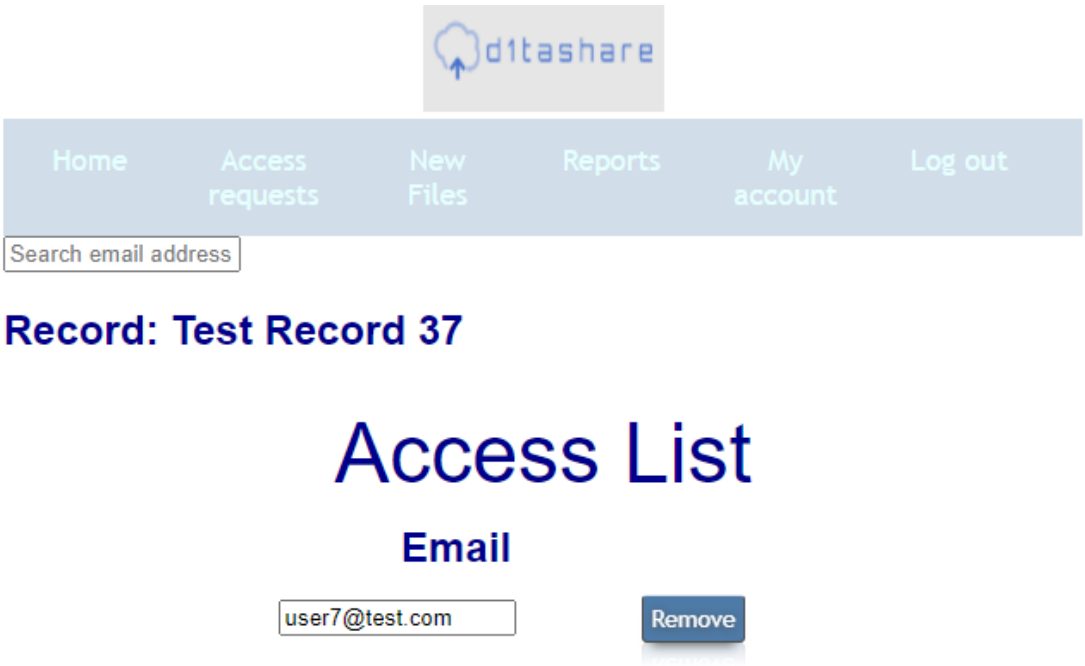
Εικόνα 31: Η αρχική σελίδα και η δυνατότητα διαχείρισης των χρηστών που έχουν πρόσβαση στο δεδομένο, επιλέγοντας το κουμπί «+» στο δεξί μέρος του κάθε δεδομένου.

Πατώντας το κουμπί «+» ο χρήστης ανακατευθύνεται στη σελίδα διαχείρισης πρόσβασης του δεδομένου, της εικόνας 32. Από εκεί μπορεί να αφαιρέσει τα παραπάνω δικαιώματα ανα πάσα στιγμή για κάθε χρήστη.



Εικόνα 32: Σελίδα διαχείρισης χρηστών με πρόσβαση στο συγκεκριμένο δεδομένο

Έστω, ο χρήστης επιλέγει να αφαιρέσει τα δικαιώματα του χρήστη με email ¨user33@test.com¨ της παραπάνω εικόνας, μέσω της επιλογής «Remove». Το αποτέλεσμα θα είναι η αφαίρεση των δικαιωμάτων του χρήστη σχετικά με το συγκεκριμένο δεδομένο και θα η λίστα με τους χρήστες που έχουν πρόσβαση στο αρχείο θα είναι αυτή της εικόνας 33.



Εικόνα 33: Η σελίδα διαχείρισης χρηστών μετά την αφαίρεση των δικαιωμάτων πρόσβασης του χρήστη

4.2.2 Αρχιτεκτονική συστήματος εφαρμογής

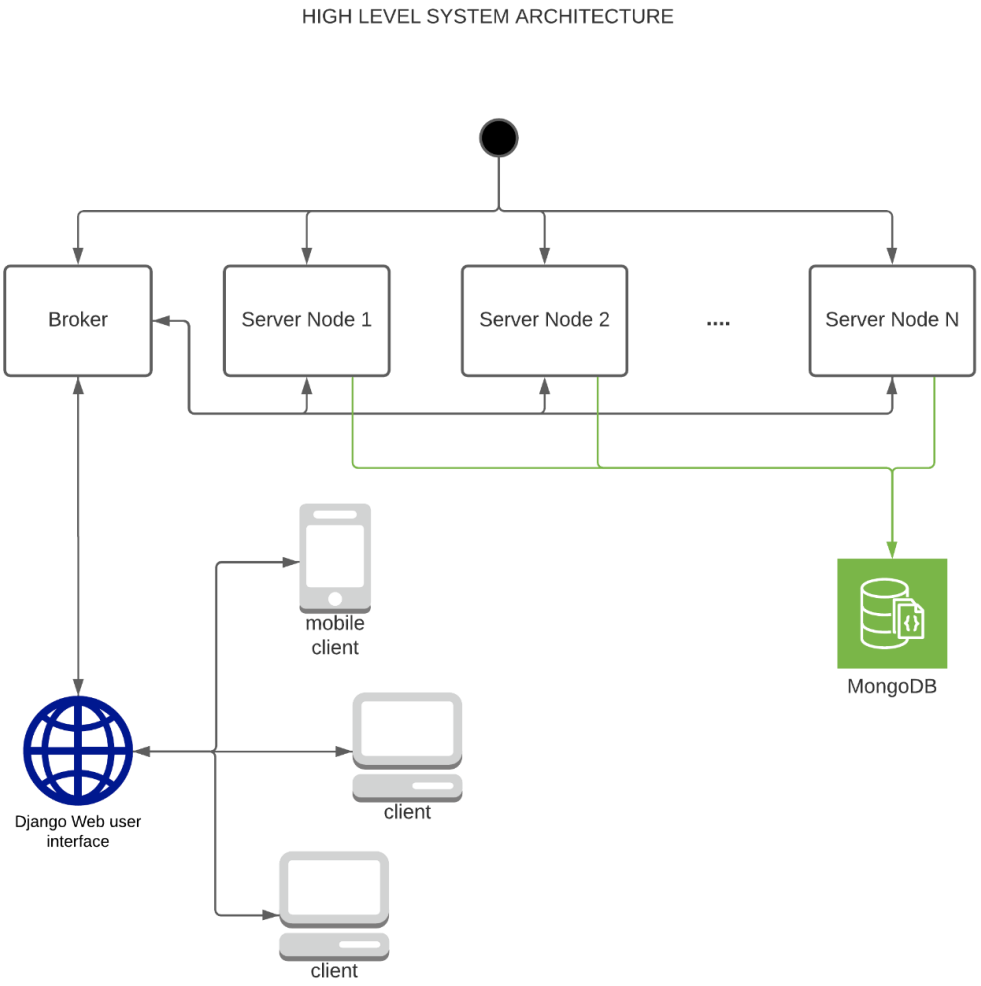
Ο στόχος της παρούσας ενότητας είναι η περιγραφή της εφαρμογής που αναπτύχθηκε κατά τα πλαίσια της συγκεκριμένης πτυχιακής εργασίας, όσον αφορά την αρχιτεκτονική του συστήματος και το γενικότερο σχεδιασμό της. Στις ενότητες που ακολουθούν πραγματοποιείται η ανάλυση των τεχνολογιών που χρησιμοποιήθηκαν, ενώ στην παρούσα θα περιγράφει ο τρόπος λειτουργίας του συστήματος σε τεχνικό επίπεδο.

Αρχικά, εφόσον πρόκειται για μια εφαρμογή όπου ο χρήστης έχει άμεση επαφή με τα δεδομένα, κρίνεται σημαντικό να γίνει ο διαχωρισμός μεταξύ της backend και frontend πλευράς της εφαρμογής. Ως backend ορίζεται οποιαδήποτε λειτουργία αφορά τη λειτουργικότητα της εφαρμογής, όπου όμως ο χρήστης δεν είναι απαραίτητο να γνωρίζει σχετικά. Αντίστοιχα, ως frontend πλευρά της εφαρμογής ονομάζεται το κομμάτι του συστήματος με το οποίο έρχεται άμεσα σε επαφή ο χρήστης, όπως είναι οι σελίδες που αναφέρθηκαν στην παραπάνω ενότητα (4.2.1 Σύντομη περιήγηση εφαρμογής και εμπειρίας του χρήστη).

Κοιτώντας την Εικόνα 30, αρχικά παρατηρείται πως το frontend κομμάτι της εφαρμογής είναι η Django web-app διεπαφή με την οποία ο χρήστης έχει τη δυνατότητα να εκτελεί όλες τις λειτουργίες που η επαφή έχει να προσφέρει. Το Django web-app είναι σε άμεση επικοινωνία με τον broker-κόμβο του συστήματος αλλά και με τον κάθε χρήστη και τα ανάλογα αιτήματα του.

Παράλληλα, γίνεται εμφανές πως ο broker-κόμβος (περιγράφεται ως «broker» στην Εικόνα 34) επικοινωνεί συνεχώς με όλους τους διαθέσιμους κόμβους (Server Node) του συστήματος. Κύριος στόχος του broker είναι η ανακατεύθυνση των αιτημάτων και αντίστοιχων απαντήσεων στο μέλος του συστήματος που το καθένα προορίζεται. Ακολουθεί ανάλυση του τρόπου λειτουργίας του broker κόμβου παρακάτω.

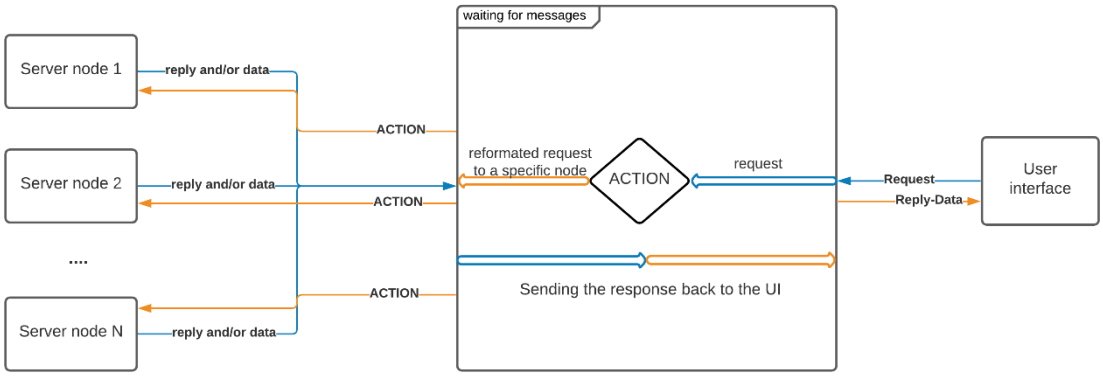
Επιπλέον, το σύστημα είναι συνδεδεμένο με μια MongoDB βάση δεδομένων που δημιουργήθηκε για τους σκοπούς της εφαρμογής και της μακροχρόνιας αποθήκευσης των δεδομένων. Αναλυτική επεξήγηση ακολουθεί σε αργότερη ενότητα επίσης.



Εικόνα 34: Συνολική αρχιτεκτονική εφαρμογής

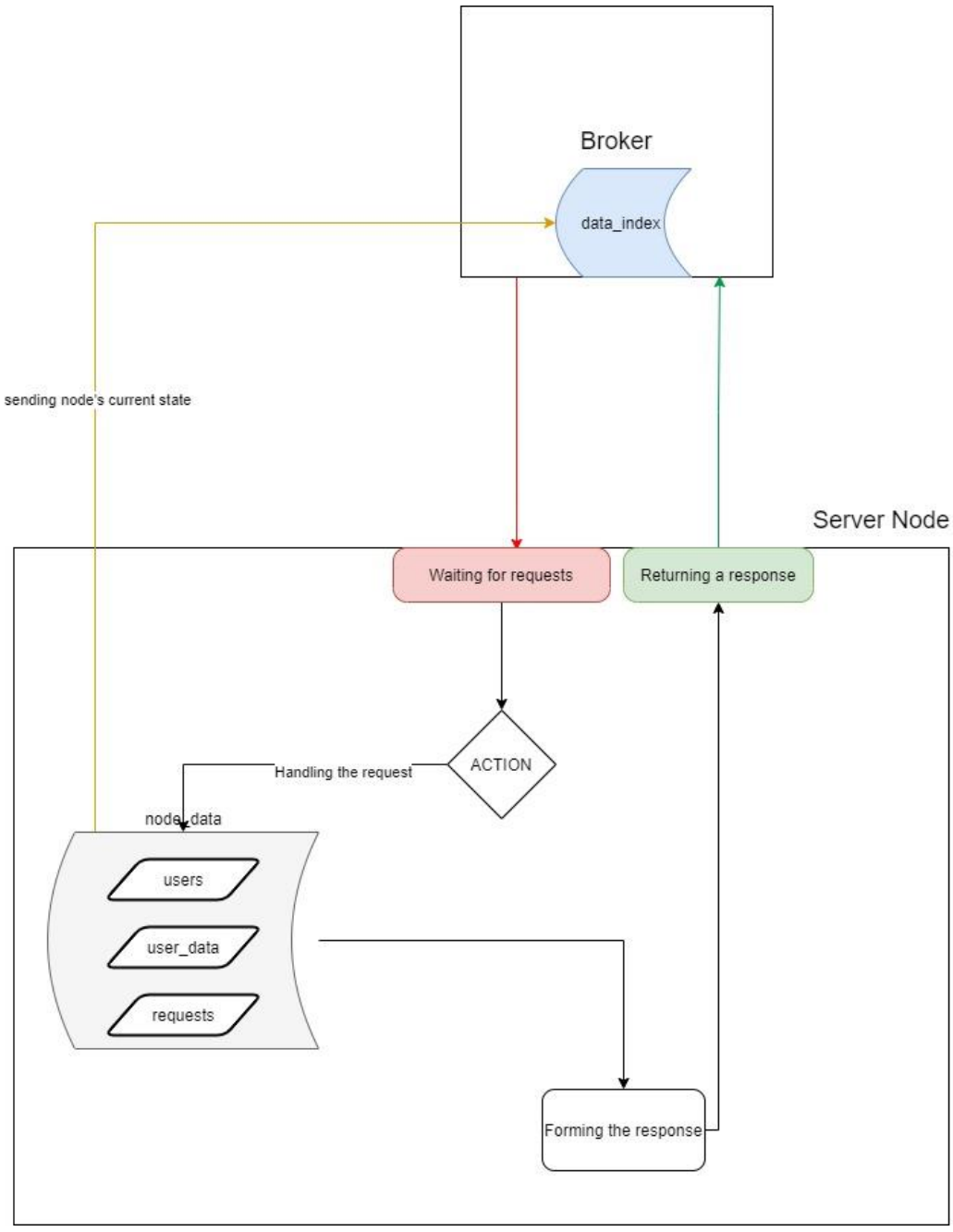
Αναλυτικότερα, ο broker κόμβος, όπως το όνομα ¨μεσίτης¨ περιγράφει, είναι υπεύθυνος για την ανακατεύθυνση των δεδομένων και αιτημάτων μεταξύ χρηστών και κόμβων. Ο κόμβος βρίσκεται μόνιμα σε κατάσταση αναμονής μηνυμάτων, έτσι ώστε να είναι σε θέση να εξυπηρετήσει όποιο αίτημα λάβει άμεσα. Όταν λαμβάνεται ένα νέο αίτημα, είναι σημαντικό να εμπεριέχεται η ανάλογη τιμή για ¨κωδικό λειτουργίας¨, όπου αναφέρεται ως «Action» στην Εικόνα 35. Η μεταβλητή Action ορίζει ποια λειτουργία ζητείται από το αίτημα του χρήστη και αναγνωρίζεται από τους Server κόμβους. Τα αιτήματα και η επιστροφή των δεδομένων στους χρήστες είναι διαδικασίες όπου πραγματοποιούνται παράλληλα και πρόκειται για διαδικασίες με ελάχιστο φόρτο εργασίας. Ο broker κόμβος είναι ενημερωμένος σχετικά με την κατάσταση των δεδομένων του κάθε server κόμβου

κάθε χρονική στιγμή. Επομένως, γνωρίζοντας την κατανομή των δεδομένων σε κάθε server κόμβο, ο broker κόμβος είναι σε θέση να ανακατευθύνει τα αιτήματα στον ανάλογο server κόμβο ιδιαίτερά αποδοτικά.



Εικόνα 35: Αρχιτεκτονική και τρόπος λειτουργίας κόμβου broker.

Τέλος, η Εικόνα 36 περιγράφει τις λειτουργίες του κάθε server κόμβου με την χρήση του σχετικού διαγράμματος.



Εικόνα 36: Διάγραμμα λειτουργίας server κόμβου

Η λειτουργία του κάθε server κόμβου μπορεί να χαρακτηριστεί σχετικά απλοϊκή, παραμένει αποτελεσματική ωστόσο. Ο κάθε κόμβος έχει τη δική του τοπική μνήμη, με όνομα «node_data» η οποία διατηρεί κατανεμημένα δεδομένα σχετικά με τις πληροφορίες των λογαριασμών τον χρηστών, τα ίδια τα δεδομένα των χρηστών και αιτήσεις

πρόσβασης. Τα συγκεκριμένα δεδομένα δεν αποτελούν αντίγραφα της βάσης δεδομένων, αλλά πρόκειται για ένα κομμάτι των πληροφοριών, καθώς πρόκειται για κατανεμημένη διαχείριση των δεδομένων. Όπως και ο broker κόμβος, ο κάθε server κόμβος περιμένει συνεχώς για πιθανά αιτήματα προς εξυπηρέτηση προερχόμενα από τον broker. Όταν κάποιο αίτημα πραγματοποιηθεί, μέσω της χρήσης της τοπικής μνήμης του κάθε κόμβου, η ανάλογη απάντηση επιστρέφεται στον broker με σκοπό την ανακατεύθυνση αυτής προς τον χρήστη. Παράλληλα, μετά από οποιαδήποτε αλλαγή, ο κάθε server κόμβος φροντίζει να ενημερώσει τον broker κόμβο, έτσι ώστε να διατηρείται η συνέπεια μεταξύ κόμβων.

4.3 Python

Αρχικά, αξίζει να αναφερθεί πως υπάρχουν πολλαπλές επιλογές όσον αφορά την επιλογή προγραμματιστικής γλώσσας για την ανάπτυξη ενός κατανεμημένου συστήματος. Η επιλογή της γλώσσας προγραμματισμού που θα χρησιμοποιηθεί για την ανάπτυξη κάποιας ιδέας είναι ένα ιδιαίτερα σημαντικό βήμα κατά το σχεδιασμό και τη δημιουργία της αρχιτεκτονικής της εφαρμογής.

Η γλώσσα προγραμματισμού Python, κατά τις τελευταίες ανανεωμένες εκδόσεις της, αποτελεί μια από τις καλύτερες επιλογές για την κάλυψη των απαιτήσεων της συγκεκριμένης εφαρμογής.

4.3.1 Python – Βασικές Πληροφορίες

Η Python είναι μια αντικειμενοστραφής, υψηλού επιπέδου γλώσσα προγραμματισμού με δυναμική σημασιολογία. Η παροχή ενσωματωμένων δομών δεδομένων υψηλού επιπέδου, σε συνδυασμό με τη χαρακτηριστική δυναμική τακτική πληκτρολόγησης και τη δυναμική δέσμευση, την καθιστούν ιδανική επιλογή όταν πρόκειται για ταχεία ανάπτυξη εφαρμογών αλλά και ως συνδετική τεχνολογία για τη σύνδεση των ήδη υπαρχόντων στοιχείων. Η έντονη προτεραιότητα στην αναγνωσιμότητα οδηγεί στη χρήση απλής και εύχρηστης σύνταξης της Python αποτελεί σημαντικό προτέρημα ενώ παράλληλα μειώνει το κόστος συντήρησης λογισμικού (What is Python? Executive Summary, n.d.).

Ακόμα, η Python υποστηρίζει τη χρήση ενοτήτων (modules) και πακέτων (packages), η οποία προωθεί τη λειτουργικότητα του προγράμματος και την επαναχρησιμοποίηση κώδικα. Αξίζει, επιπλέον, να γίνει αναφορά αναφερθούμε στο διερμηνέα (interpreter) της Python και στην εκτεταμένη τυπική βιβλιοθήκη, όπου είναι διαθέσιμες σε πολλαπλές

μορφές, χωρίς χρέωση, υποστηρίζεται από όλες τις μεγάλες πλατφόρμες και τέλος μπορούν να κατανεμηθούν ελεύθερα.

Η Python επιλέγεται συχνά από προγραμματιστές λόγω της αυξημένης παραγωγικότητας που προσφέρει. Λαμβάνοντας υπ' όψιν ότι δεν υπάρχει το ανάλογο βήμα μεταγλώττισης, όπου πολλές γλώσσες προγραμματισμού απαιτούν για τη λειτουργία τους, η διαδικασία δοκιμής και εντοπισμού σφαλμάτων είναι σημαντικά γρήγορη. Γενικότερα, ο εντοπισμός σφαλμάτων σε προγράμματα Python είναι ευκολότερος, ενώ παράλληλα ένα σφάλμα ή μια λάθος είσοδος δεν θα προκαλέσει σφάλμα κατάτμησης (segmentation fault). Αντί αυτού, όταν ο διερμηνέας ανακαλύψει ένα σφάλμα, δημιουργεί μια εξαίρεση (exception). Στις περιπτώσεις όπου η εφαρμογή αποτυγχάνει να χειριστεί την εξαίρεση, διαχειρίζεται την κατάσταση παράγοντας ένα σχετικό μήνυμα λάθους.

Ο εντοπισμός σφαλμάτων σε επίπεδο πηγής είναι ένα εργαλείο που επιτρέπει τον εντοπισμό σφαλμάτων ακόμα και σε περιπτώσεις πολύπλοκου κώδικα. Έχει τη δυνατότητα να εκτελεί αξιολόγηση διαφόρων τύπων εκφράσεων, καθορισμό σημείων διακοπής, τη διέλευση του κώδικα από μια γραμμή κάθε φορά και ούτω καθεξής.

4.3.2 Χαρακτηριστικά της γλώσσας προγραμματισμού Python

Είναι κοινά παραδεκτό ότι η Python είναι εύκολη για εκμάθηση και χρήση. Παρόλο που πρόκειται για μια γλώσσα υψηλού επιπέδου με πολλαπλές πηγές για μάθηση και υποστήριξη μιας μεγάλης ποικιλίας σχετικών εργαλείων καθιστούν ευκολότερη τη χρήση της παρούσας γλώσσας προγραμματισμού και οι χρήστες παρακινούνται να συνεχίσουν. Η σύνταξη της Python χαρακτηρίζεται ως ιδιαίτερα απλή αλλά και κομψή. Ο κώδικας σε Python είναι πιο ευανάγνωστος συγκριτικά με άλλες γλώσσες προγραμματισμού όπως C ++, Java και C#, γεγονός που καθιστά τον προγραμματισμό σε Python πιο ευχάριστο και φιλικό για αρχάριους. (Srinath, 2017)

Επιπροσθέτως η Python είναι φορητή. Ο κώδικας σε Python έχει τη δυνατότητα να εκτελείται σε διάφορα λειτουργικά συστήματα, συμπεριλαμβανομένων των Windows, Linux, UNIX, Amigo και Mac OS. Οι εφαρμογές Python μπορούν να μετακινηθούν ανάμεσα σε πλατφόρμες και να εκτελεστεί επιτυχώς χωρίς τροποποιήσεις.

Παράλληλα ένα σημαντικό χαρακτηριστικό είναι πως η Python κατηγοριοποιείται ως ανοιχτός κώδικας. Αν και το Python Institute διατηρεί όλα τα δικαιώματα της γλώσσας

προγραμματισμού, είναι ανοιχτού κώδικα και μπορεί να χρησιμοποιηθεί και να διανεμηθεί χωρίς περιορισμούς. Η Python είναι ελεύθερη για χρήση και διανομή δωρεάν, ακόμη και για εμπορικούς σκοπούς. Είναι ακόμα δυνατό όχι μόνο να χρησιμοποιηθούν και να διανεμηθούν εφαρμογές που βασίζονται σε Python, αλλά επίσης να τροποποιηθεί ο πηγαίος κώδικας της Python. Συγκεκριμένα η Python κατέχει μια μεγάλη κοινότητα που τη βελτιώνει συνεχώς.

Αξίζει ακόμη, να τονιστεί ότι η Python υποστηρίζει άλλες τεχνολογίες όπως COM, .Net, κλπ. Έτσι μπορεί να επιτευχθούν υψηλές αποδόσεις σε σύγκριση με άλλες γλώσσες προγραμματισμού.

Ακόμα ένα χαρακτηριστικό της γλώσσας είναι πως πρόκειται για μια γλώσσα υψηλού επιπέδου, το οποίο από τη φύση του προσφέρει κάποια πλεονεκτήματα. Σε αντίθεση με άλλες γλώσσες, όπως η C/C++, εργασίες όπως η διαχείριση και καθαρισμός της μνήμης έχουν απλοποιηθεί σε μεγάλο βαθμό. Ανάλογα, κατά την εκτέλεση κώδικα Python η μετατροπή του κώδικα στη γλώσσα που απαιτείται από τα χαμηλότερα επίπεδα του υπολογιστικού συστήματος συμβαίνει αυτόματα.

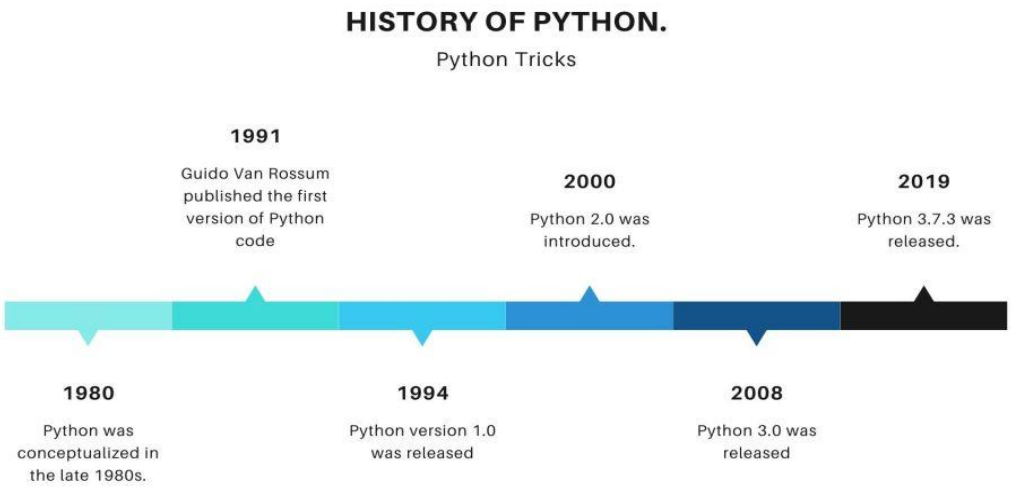
4.3.3 Εκδόσεις Python

Σε αντίθεση με άλλες δημοφιλείς γλώσσες προγραμματισμού, η Python θέσπισε δύο διαφορετικές εκδόσεις της, οι οποίες διαφέρουν σημαντικά μεταξύ τους. Χρονικά συνέβη ως εξής (Python Documentation by Version, 2021):

1. Η έκδοση Python 2.0 δημοσιεύθηκε στις 16 Οκτωβρίου 2000 από τη BeOpen.com, το Guido van Rossum και άλλους PythonLabs προγραμματιστές.
2. Η μεταγενέστερη έκδοση Python 2.7 (4 Ιουλίου 2010) παρέμεινε ως η βασική έκδοση για ένα μεγάλο χρονικό διάστημα.
3. Η Python 3 δημοσιεύτηκε στις 3 Δεκεμβρίου 2008 αλλά λόγω κάποιων βασικών αλλαγών που παρουσιάστηκαν στη γλώσσα δεν υιοθετήθηκε γρήγορα από τους χρήστες.
4. Παράλληλα ενώ η έκδοση Python 2.7 εξακολουθεί να χρησιμοποιείται αποφασίστηκε πως θα λαμβάνει τις απαραίτητες ενημερώσεις ασφαλείας έως και το 2020.

5. Η έκδοση Python 3 μετατράπηκε στην κύρια επιλογή των προγραμματιστών και εξελίσσεται συνεχώς όπως η Python 2 στο παρελθόν.

Ακολουθεί σχετική Εικόνα 37 όπου απεικονίζεται η πορεία της γλώσσας Python με ένα σχετικό χρονοδιάγραμμα.



Εικόνα 37: History of Python - Topictrick.com March 2020

Για τη ανάπτυξη της σχετικής εφαρμογής που αναπτύχθηκε για την παρούσα εργασία επιλέχθηκε η έκδοση Python 3 και πιο συγκεκριμένα η έκδοση Python 3.8.
Ενώ οι δύο εκδόσεις παρουσιάζουν πολλαπλές διαφορές, παρακάτω αναφέρονται οι βασικοί λόγοι που επιλέχθηκε η συγκεκριμένη έκδοση:

- Η πλειοψηφία των χρηστών επιλέγουν Python 3, κατατάσσοντας την έκδοση Python 2 στο παρελθόν.
  Όπως είναι αναμενόμενο, όντας η πιο δημοφιλής έκδοση για τουλάχιστον μια δεκαετία, η Python 2 εξακολουθεί να είναι το παγιωμένο λογισμικό που χρησιμοποιείται σε ορισμένους χώρους. Ωστόσο, τα τελευταία χρόνια παρατηρείται πως οι περισσότερες εταιρείες και οργανισμοί μετακινούνται από τη Python 2 στη 3. Επίσης όπως αναφέρθηκε ήδη, ο οργανισμός της Python δήλωσε πως δεν θα συνεχίσει να δημιουργεί ανάλογες ενημερώσεις ασφαλείας για την

παλαιότερη έκδοση, γεγονός που δηλώνει την ενθάρρυνση των χρηστών προς την υιοθέτηση της νεότερης έκδοσης.

- Διαφορετικές και πιθανόν ασύμβατες βιβλιοθήκες μεταξύ εκδόσεων.
  Εφόσον η Python 3 είναι η έκδοση του μέλλοντος πολλοί προγραμματιστές δημιουργούν βιβλιοθήκες αποκλειστικά για την τελευταία έκδοση. Παρόμοια, πολλές από τις βιβλιοθήκες που δημιουργήθηκαν για την έκδοση Python 2 δεν είναι συμβατές με τις νεότερες εκδόσεις.
- Καλύτερη υποστήριξη Unicode στη Python 3.
  Επειδή το σύστημα Unicode είναι πιο ευέλικτο από το ASCII, αυτή η διαφορά είναι κρίσιμη. Οι συμβολοσειρές Unicode μπορούν να διατηρήσουν γράμματα από άλλες γλώσσες, ρωμαϊκά γράμματα και αριθμούς, σύμβολα, emojis, κτλ. προσφέροντας έτσι επιπλέον επιλογές.
- Η Python 3 έχει βελτιώσει τη διαίρεση ακέραιων αριθμών.

4.4 Multithreading και Multiprocessing σε Python

Η Python ως μια δημοφιλής γλώσσα προγραμματισμού τα τελευταία χρόνια χρησιμοποιείται συνήθως σε διάφορους τομείς εμπορικούς και επιστημονικούς. Λόγω του μεγάλου εύρους εφαρμογών του, απαιτείται από τη γλώσσα προγραμματισμού να μπορεί να λειτουργεί ομαλά σε σύγχρονο υλικό με τις ανάλογες απαιτήσεις παράλληλου υπολογισμού. Για αυτό το λόγο η Python διαθέτει δύο τεχνικές για την ανάπτυξη εφαρμογών με τη χρήση παραλληλισμού, την πολυεπεξεργασία (multiprocessing) και την πολυνηματική επεξεργασία (multithreading) (Meier & Gross, 2019).

4.4.1 Multiprocessing με Python

Αρχικά είναι σημαντικό να αναφερθεί πως χωρίς τη χρήση πολλαπλής επεξεργασίας, οι εφαρμογές τις Python θα περιοριζόταν και δεν θα είχαν τη δυνατότητα να μεγιστοποιήσουν τις συνολικές προδιαγραφές του συστήματος. Η Python δεν κατασκευάστηκε εξαρχής με γνώμονα τη δυνατότητα χρήσης πολλών πυρήνων, επομένως επιπρόσθετα εργαλεία απαιτούνται, καθώς η Python δεν είναι ασφαλής για νήματα και την παράλληλη διαχείριση αντικειμένων Python. Ωστόσο, η τεχνική πολλαπλής επεξεργασίας είναι ένας αρκετά αποτελεσματικός μηχανισμός για τη διαχείριση της μνήμης κατά την ανάπτυξη κατανεμημένων-παράλληλων συστημάτων.

Η πολυεπεξεργασία είναι μια τεχνική όπου επιτυγχάνεται ο παραλληλισμός στην πιο αληθινή του μορφή επιτρέποντας τη δημιουργία προγραμμάτων που έχουν τη δυνατότητα να εκτελούνται ταυτόχρονα και τη χρήση ολόκληρου του πυρήνα της CPU. Πολλαπλές διαδικασίες εκτελούνται σε πολλούς πυρήνες CPU, οι οποίοι δεν μοιράζονται τους πόρους μεταξύ τους. Κάθε διαδικασία μπορεί να έχει πολλά νήματα που τρέχουν στο δικό της χώρο μνήμης (Varun, 2020). Η σύνταξη είναι αρκετά παρόμοια με τη βιβλιοθήκη νημάτων, παρά το γεγονός ότι είναι θεμελιωδώς διαφορετική. Κάθε διαδικασία έχει το δικό της Python διερμηνέα και GIL χάρη στη βιβλιοθήκη πολλαπλής επεξεργασίας. Τα συνήθης ζητήματα κατά τη χρήση νημάτων (όπως η καταστροφή δεδομένων και τα αδιέξοδα) δεν αποτελούν πλέον ανησυχία ως αποτέλεσμα αυτού. Επειδή οι διαδικασίες δεν μοιράζονται μνήμη, δεν μπορούν να αλλάξουν την ίδια μνήμη ταυτόχρονα (Panjwani, 2018).

Συνοπτικά τα κύρια χαρακτηριστικά της πολλαπλής επεξεργασίας ως διαδικασία είναι τα εξής:

- Οι πολυεπεξεργασίες ταξινομούνται ανάλογα με τον τρόπο οργάνωσης της μνήμης τους.
- Η πολλαπλή επεξεργασία βελτιώνει την αξιοπιστία του συστήματος.
- Η πολυεπεξεργασία μπορεί να βελτιώσει την απόδοση αποσυνθέτοντας ένα πρόγραμμα σε παράλληλες μικρότερες εκτελέσιμες εργασίες.

4.4.2 Multithreading με Python

Συνοπτικά, μέσα σε μια διαδικασία, ένα νήμα είναι μια μονάδα εκτέλεσης. Έτσι το Multithreading είναι η διαδικασία εκτέλεσης πολλών νημάτων ταυτόχρονα, εναλλάσσοντας γρήγορα τον έλεγχο της CPU μεταξύ τους (ονομάζεται εναλλαγή περιβάλλοντος) (Ziadé, 2008). Όταν διακόπτεται ένα νήμα (λόγω εισόδου/εξόδου ή ρητής ρύθμισης), η κατάσταση αυτού του νήματος διατηρείται και η κατάσταση ενός άλλου νήματος φορτώνεται. Η εναλλαγή περιβάλλοντος συμβαίνει τόσο συχνά που όλα τα νήματα φαίνεται να εκτελούνται ταυτόχρονα (αυτό το φαινόμενο ονομάζεται πολυνηματικές διεργασίες).

Γενικότερα το multithreading είναι μια πολύ γνωστή μέθοδος στον προγραμματισμό Python. η οποία επιτρέπει σε πολλά νήματα σε μια διαδικασία να μοιράζονται τον αποθηκευτικό χώρο τους με το κύριο νήμα, καθιστώντας την ανταλλαγή πληροφοριών και την επικοινωνία μεταξύ των νημάτων απλή και αποτελεσματική. Οι διαδικασίες χαρακτηρίζονται ως βαρύτερες από τα νήματα. Επίσης είναι δυνατό πολλαπλά νήματα να

εκτελούνται παράλληλα ενώ μοιράζονται τους πόρους της διαδικασίας. Ο γενικότερος στόχος του multithreading είναι να εκτελούνται πολλές εργασίες ταυτόχρονα.

Παρακάτω αναφέρονται μερικές από τις σημαντικές πτυχές της διαδικασίας πολλαπλών νημάτων:

- Στη διαδικασία πολλαπλών νημάτων, κάθε νήμα τρέχει παράλληλα μεταξύ των υπολοίπων.
- Τα νήματα δεν επιτρέπουν το διαχωρισμό της περιοχής μνήμης. Ως εκ τούτου, εξοικονομεί μνήμη και προσφέρει καλύτερη απόδοση εφαρμογής.

4.4.3 Multithreading και Multiprocessing - διαφορές

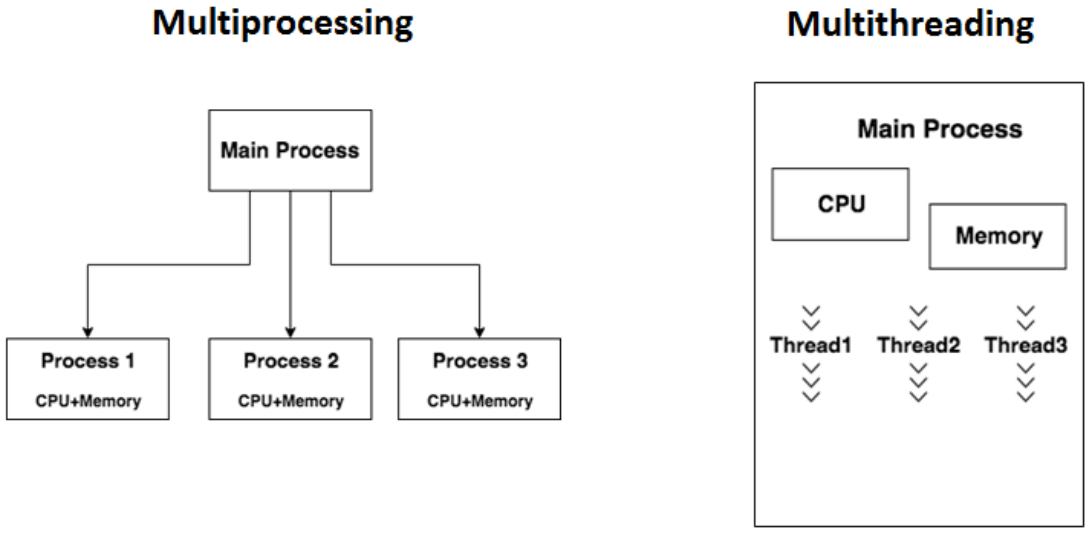
Παρακάτω αναφέρονται οι βασικές διαφορές μεταξύ των δύο τεχνικών παράλληλου υπολογισμού με βάση τον τομέα που επηρεάζουν (Multithreading vs Multiprocessing: What's the difference?, 2020)):

- **Συνολικός χρόνος εκτέλεσης:** Η διαδικασία της πολλαπλής επεξεργασίας απαιτεί λιγότερο χρόνο για την εκτέλεση μιας διεργασίας σε σύγκριση με τη διαδικασία πολλαπλών νημάτων που η ολοκλήρωση των εργασιών συνήθως χρειάζεται ένα εύλογο χρονικό διάστημα.
- **Εκτέλεση:** Η πολυεπεξεργασία επιτρέπει την ταυτόχρονη εκτέλεση πολλαπλών διαδικασιών. Από την άλλη πλευρά κατά την τεχνική multithreading πολλαπλά νήματα μιας μεμονωμένης διαδικασίας εκτελούνται ταυτόχρονα.
- **Υποστηριζόμενα προγράμματα και εργασίες:** Το σύστημα πολλαπλών νημάτων εκτελεί πολλά νήματα της ίδιας ή διαφορετικής διαδικασίας. Ενώ ένα σύστημα πολλαπλής επεξεργασίας επιτρέπει την εκτέλεση πολλαπλών προγραμμάτων και εργασιών.
- **Εναλλαγή CPU:** Για να φαίνεται ότι όλα τα νήματα εκτελούνται ταυτόχρονα, η CPU πρέπει να κάνει κύκλους μεταξύ διαφόρων νημάτων κατά τη χρήση πολλαπλών νημάτων. Αντίστοιχα στη μέθοδο της πολυεπεξεργασίας, η CPU πρέπει να αλλάζει μεταξύ πολλαπλών προγραμμάτων, έτσι ώστε να φαίνεται ότι εκτελούνται ταυτόχρονα πολλά προγράμματα
- **Διατήρηση αντικειμένων:** Κατά τη χρήση πολλαπλών παράλληλων νημάτων δεν διατηρούνται άσκοπα αντικείμενα και πληροφορίες. Απεναντίας, η πολλαπλή

επεξεργασία βασίζεται στην αποθήκευση αντικειμένων στη μνήμη για αποστολή σε άλλες διαδικασίες.

- **Δημιουργία νέων πόρων/νημάτων/διαδικασιών:** Η δημιουργία μιας διαδικασίας μπορεί είναι αργή και να έχει συγκεκριμένες απαιτήσεις από τους πόρους του συστήματος. Σε αντίθεση με τη δημιουργία ενός νήματος που είναι οικονομική σε χρόνο και πόρους.
- **Χρήση της μνήμης:** Τα multithreading νήματα στην ίδια διαδικασία έχουν πρόσβαση στην ίδια μνήμη και πόρους με την ίδια τη διαδικασία επομένως έχουν αναπτυχθεί ανάλογες τεχνικές για την ομαλή κατανομή της μνήμης μεταξύ νημάτων. Σε πλήρη αντιδιαστολή η πολλαπλή επεξεργασία διαθέτει ξεχωριστή μνήμη και πόρους για κάθε διαδικασία ή πρόγραμμα.
- Ταξινόμηση: Η πολλαπλή επεξεργασία μπορεί να είναι συμμετρική ή ασύμμετρη ενώ διαδικασία multithreading δεν είναι ταξινομημένη.
- Τέλος μια γενική διαφορά μεταξύ των τεχνικών έχει να κάνει με το διαφορετικό τρόπο (άμεσο ή έμμεσο) που η καθεμιά συμβάλλει στη βελτίωση της υπολογιστικής ισχύος. Η πολλαπλή επεξεργασία βοηθά στη γενικότερη αύξηση της υπολογιστικής ισχύος ενώ η χρήση πολλαπλών νημάτων βοηθά στη δημιουργία υπολογιστικών νημάτων μιας μεμονωμένης διαδικασίας για την αύξηση της υπολογιστικής ισχύος.

Παρατηρώντας τους διαφορετικούς τρόπους λειτουργίας μεταξύ των τεχνικών στην Εικόνα 38 γίνονται αντιληπτές οι βασικές διαφορές.



Εικόνα 38: Multithreading vs Multiprocessing in Python- Varun Kumar, 2020

4.5 Django

Ο Adrian Holovaty και ο Simon Willison άρχισαν να χρησιμοποιούν τη Python για τη δημιουργία εφαρμογών το 2003 και η τεχνολογία Django γεννήθηκε. Η ανάγκη ανάπτυξης εφαρμογών σε καθημερινή ή ακόμη και ωριαία βάση οδήγησε στην ανάπτυξη του Django. Δημοσιεύτηκε ως εργαλείο ανοιχτού κώδικα το 2005, δύο χρόνια αφότου άρχισε να αναπτύσσει ένα πλαίσιο εφαρμογών Ιστού (Web application framework). Το Django έχει τώρα δεκάδες χιλιάδες χρήστες και συνεχίζει να αυξάνεται (Dauzon, Bendoraitis, & Ravindran, 2016). Όπως αναφέρθηκε παραπάνω, το Django είναι ένα web application framework που βασίζεται στη γλώσσα προγραμματισμού Python, επομένως θα χρειαστεί η χρήση Python για να το χρήση του. Παρόλο που η Python περιλαμβάνει έναν υποτυπώδες διακομιστή δοκιμών, ο μελλοντικός ιστότοπος που βασίζεται στο Django θα απαιτεί διακομιστή Apache και το πακέτο mod wsg για να λειτουργεί σωστά. Ακόμα το Django υποστηρίζει επίσημα βάσεις δεδομένων PostgreSQL, MySQL, Oracle και SQLite (Ghimire, 2020).

Το Django διατηρεί την προέλευσή του ως framework διακομιστής Model-View-Controller (MVC) σχεδιασμένο να λειτουργεί με σχεσιακές βάσεις δεδομένων. Το Django, από την άλλη πλευρά, έχει συμβαδίσει με τις περισσότερες τάσεις ανάπτυξης ιστοσελίδων, χρησιμοποιώντας εξωτερικά πακέτα (third-party packages) για την υποστήριξη τεχνολογιών όπως μη σχεσιακές βάσεων δεδομένων (NoSQL), συνδεσιμότητα σε πραγματικό χρόνο στο Διαδίκτυο και σύγχρονες τεχνικές JavaScript.

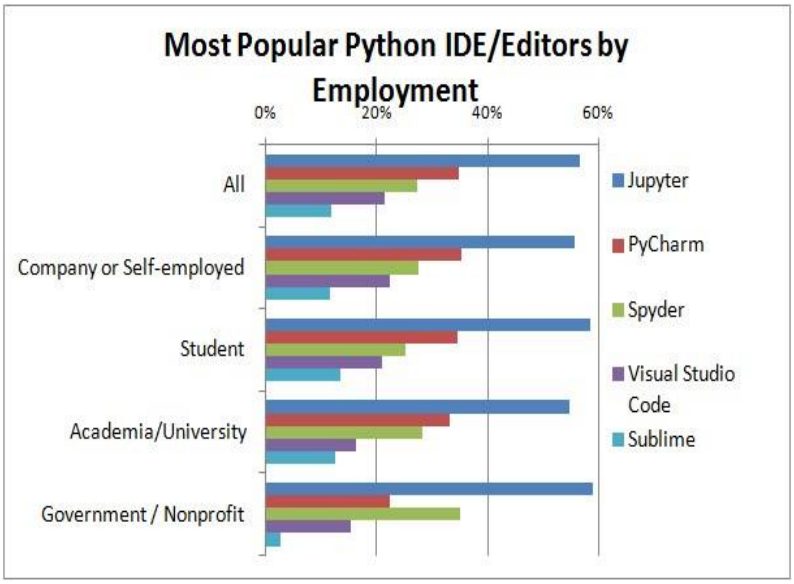
Συνοψίζοντας, το Django είναι πλέον το εργαλείο ανάπτυξης ιστοσελίδων για ένα ευρύ φάσμα επιχειρήσεων, συμπεριλαμβανομένων των υπηρεσιών κοινής χρήσης φωτογραφιών Instagram και Pinterest, του Public Broadcasting System (PBS) των Ηνωμένων Πολιτειών, του National Geographic και άλλων (Rubio, 2017).

4.6 Το Περιβάλλον Ανάπτυξης - PyCharm

Απαραίτητη κρίνεται η αναφορά και ο ορισμός των Ολοκληρωμένων Περιβαλλόντων Ανάπτυξης ή IDE. Ένα IDE διαθέτει έναν επεξεργαστή κώδικα (code editor) και έναν μεταγλωττιστή για τη σύνταξη και τη σύνταξη προγραμμάτων σε μία ή περισσότερες

γλώσσες προγραμματισμού. Επιπλέον, ένα IDE διαθέτει πληθώρα δυνατοτήτων που βοηθούν στην πλήρη ανάπτυξη λογισμικού. Αναλύει την πληκτρολόγηση και προσφέρει την κατάλληλη λέξη -κλειδί για χρήση. Γίνεται εμφανής η διαφορά μεταξύ μιας κλάσης και μιας μεθόδου επειδή το IDE κάνει χρήση ξεχωριστών χρωμάτων. Το IDE χρησιμοποιεί επίσης διάφορα χρώματα για να υποδείξει ποιες λέξεις -κλειδιά είναι σωστές και ποιες εσφαλμένες. Εάν ο χρήστης πληκτρολογήσει κάποια λανθασμένη λέξη -κλειδί, το IDE προσπαθεί να μαντέψει τη λέξη -κλειδί που ο χρήστης σχεδίαζε να πληκτρολογήσει και τη συμπληρώνει αυτόματα (What is PyCharm?, 2021).

Το PyCharm είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης προγραμματισμού υπολογιστών (IDE) που ειδικεύεται στη γλώσσα προγραμματισμού Python και πρόκειται για μια από τις πιο δημοφιλής επιλογές Python IDE τα τελευταία χρόνια (Εικόνα 39). Η JetBrains, μια τσεχική εταιρεία, τη δημιούργησε (παλαιότερα γνωστή ως IntelliJ). Επίσης το PyCharm είναι πολλαπλή πλατφόρμα, με εκδόσεις Windows, macOS και Linux. Παρέχει ένα εξαιρετικό περιβάλλον για παραγωγική ανάπτυξη της επιστήμης Python, ιστού και δεδομένων, ενσωματώνοντας στενά ένα ευρύ φάσμα βασικών εργαλείων για προγραμματιστές της Python (ανάλυση κώδικα, γραφικός εντοπισμός σφαλμάτων (debugger), ενσωματωμένος ελεγκτής μονάδων (unit tester), ενσωμάτωση με συστήματα ελέγχου έκδοσης (VCSes), ανάπτυξη ιστοσελίδων με το Django, ανάλυση δεδομένων με την Anaconda).



Εικόνα 39: Δημοφιλείς Python IDEs / Editors, KDnuggets – 2018

Συγχρόνως, το PyCharm προσφέρει τρεις εκδόσεις: Community, Professional και Edu, όπου η κάθε μια καλύπτει διαφορετικές απαιτήσεις και υπηρεσίες. Όσον αφορά τις Python εκδόσεις, υποστηρίζονται εξίσου. Για ανάπτυξη σε Python 2 επιτρέπεται η χρήση της έκδοσης 2.7 και για την έκδοση Python 3 υποστηρίζονται οι εκδόσεις 3.6 μέχρι και 3.10 (Get started, 2021).

Για την ανάπτυξη της παρούσας εργασίας χρησιμοποιήθηκε η έκδοση PyCharm 2021.1.3 Professional και όπως προαναφέρθηκε Python 3.8.

4.7 Mongo DB - Διαχείριση Δεδομένων εφαρμογής – Σχήμα βάσης δεδομένων

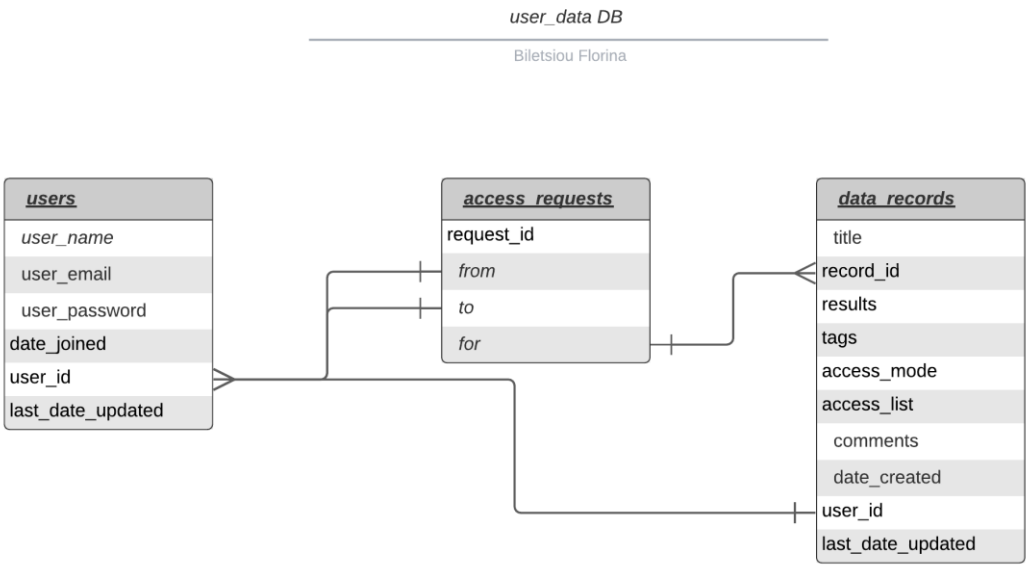
4.7.1 Mongo DB

Στα πλαίσια της ανάπτυξης της σχετικής εφαρμογής χρησιμοποιήθηκε η βάση δεδομένων που παρέχεται από την εταιρεία Mongo. Συγκεκριμένα εκτελεί το ρόλο της μόνιμης αποθήκευσης των δεδομένων, που είναι ένα σημαντικό χαρακτηριστικό για μια ολοκληρωμένη εφαρμογή. Ωστόσο το σύστημα που αναπτύχθηκε εστιάζει περισσότερο στην τοπική μνήμη του κάθε κόμβου και στην ταχεία εξυπηρέτηση μέσω της συγκεκριμένης εφαρμογής. Επομένως αποφασίστηκε πως για λόγους πληρότητας θα γίνει μια σύντομη αναφορά στην τεχνολογία που χρησιμοποιήθηκε καθώς και στη δομή των δεδομένων που διατηρούν τα δεδομένα της εφαρμογής.

Το MongoDB είναι ένα πρόγραμμα document-oriented βάσης δεδομένων που χρησιμοποιείται για αποθήκευση δεδομένων μεγάλου όγκου. Κατατάσσεται ως πρόγραμμα βάσης δεδομένων NoSQL, το MongoDB χρησιμοποιεί έγγραφα τύπου JSON με προαιρετικά σχήματα. Το MongoDB έχει αναπτυχθεί από τη MongoDB Inc. και έχει άδεια υπό Server Side Public License (SSPL) (What Is MongoDB?, 2021). Η βάση δεδομένων MongoDB χρησιμοποιεί συλλογές και έγγραφα και όχι πίνακες και σειρές όπως στις παραδοσιακές σχεσιακές βάσεις δεδομένων. Τα έγγραφα αποτελούνται από ζεύγη κλειδιού-τιμής, τα οποία αποτελούν τη θεμελιώδη μονάδα δεδομένων στο MongoDB. Οι συλλογές είναι ισοδύναμες με πίνακες σχεσιακής βάσης δεδομένων, καθώς περιλαμβάνουν συλλογές εγγράφων και λειτουργιών.

4.7.2 Διαχείριση Δεδομένων εφαρμογής – Σχήμα βάσης δεδομένων

Κατά τη δημιουργία της σχετικής κατανεμημένης εφαρμογής, η οποία χαρακτηρίζεται από υψηλά επίπεδα διαθεσιμότητας, χρησιμοποιήθηκε το παρακάτω σχήμα βάσης δεδομένων, όπου εμφανίζεται στην Εικόνα 40, για τη μακροπρόθεσμη αποθήκευση των δεδομένων:



Εικόνα 40: Αρχιτεκτονική βάσης δεδομένων MongoDB

Όπως έχει ήδη αναφερθεί, χρησιμοποιώντας τη βάση δεδομένων Mongo DB αρχικά πραγματοποιείται η δημιουργία μιας βάσης δεδομένων (user_data) και στη συνέχεια δημιουργούμε διάφορες συλλογές εντός της βάσης δεδομένων (users, access_requests, data_records).

Κοιτώντας τις συλλογές πιο αναλυτικά:

- **users**
    - **user_name**: Όνομα χρήστη το οποίο επιλέγεται κατά την εγγραφή. Μπορεί να είναι το πλήρες ονοματεπώνυμο ή μια λέξη.
    - **user_email**: Διεύθυνση ηλεκτρονικού ταχυδρομείου χρήστη.
    - **user_password**: Κωδικός που επιλέγεται από το χρήστη κατά την εγγραφή. Αποθηκεύεται κρυπτογραφημένος για λόγους ασφάλειας.
    - **date_joined**: Ημερομηνία εγγραφής του χρήστη. Συμπληρώνεται αυτόματα κατά τη διαχείριση της εγγραφής στο σύστημα.

- **user_id**: μοναδική αλληλουχία αναγνώρισης (αριθμοί και γράμματα) που ανατίθεται για κάθε χρήστη από το σύστημα κατά τη δημιουργία του λογαριασμού του.
- **last_date_updated**: Πιο πρόσφατη ημερομηνία που το έγγραφο του χρήστη ανανεώθηκε με κάποια νέα πληροφορία.

- **data_records**
  - **title**: Τίτλος αρχείου. Ο χρήστης είναι υπεύθυνος για την επιλογή του τίτλου. (υποχρεωτικό)
  - **record_id**: μοναδική αλληλουχία αναγνώρισης (αριθμοί και γράμματα) που ανατίθεται για κάθε αρχείο από το σύστημα κατά την εισαγωγή του στο σύστημα. (υποχρεωτικό)
  - **results**: Η τιμή του αρχείου. Η παρούσα πληροφορία είναι το κεντρικό κομμάτι της εφαρμογής. Αριθμητικές τιμές μόνο. (υποχρεωτικό)
  - **tags**: Πίνακας με λέξεις κλειδιά που ο χρήστης θεωρεί πως περιγράφουν το αρχείο. Σκοπός τους είναι η μετέπειτα ταξινόμηση ή εντοπισμός ενός αρχείο μέσω των λέξεων - κλειδιά.
  - **access_mode**: (υποχρεωτικό) κωδικός λειτουργίας πρόσβασης που θέτει ο ιδιοκτήτης του αρχείου. Επιτρεπτές τιμές αποτελούν οι εξής:
    - G (green) = Δημόσια πρόσβαση. Άλλοι χρήστες έχουν το δικαίωμα πρόσβασης στο αρχείο χωρίς να απαιτείται άδεια από τον ιδιοκτήτη του αρχείου.
    - Y (yellow) = Περιορισμένη πρόσβαση. Ορισμένοι χρήστες έχουν δικαίωμα πρόσβασης αφότου αποτελείται και γίνεται αποδεκτό το ανάλογο αίτημα πρόσβασης.
    - R (red) = Αυστηρά ιδιωτική πρόσβαση από μόνο τον ιδιοκτήτη του αρχείου. Δεν δίνεται η δυνατότητα πρόσβασης σε κανένα χρήστη του συστήματος πέρα από τον ιδιοκτήτη.
  - **access_list**: Πίνακας με την αναγνωριστική αλληλουχία (user_id) όλων των χρηστών που έχουν αποκτήσει πρόσβαση στο αρχείο μετά από την ανάλογη αίτηση. Ο ιδιοκτήτης του αρχείου προστίθεται πάντα στη λίστα πρόσβασης του αρχείου καθώς εννοείται πως του επιτρέπεται η πρόσβαση στο αρχείο που εισήγαγε στο σύστημα.

- **comments**: πίνακας με σχόλια σχετικά με το αρχείο. Σκοπός των σχολίων είναι να διατηρηθούν πιθανές παρατηρήσεις οι οποίες σχετίζονται με το αρχείο και να βοηθούν στην κατανόηση του αποτελέσματος.
- **date_created**: Ημερομηνία εισαγωγής του αρχείου. Συμπληρώνεται αυτόματα κατά τη διαχείριση της εισαγωγής στο σύστημα.
- **user_id**: Αναγνωριστική αλληλουχία του χρήστη ο οποίος είναι ο ιδιοκτήτης του αρχείου. Συνδετική τιμή με την ανάλογη ομότιτλη στήλη της συλλογής των χρηστών (users). (υποχρεωτικό)
- **last_date_updated**: Πιο πρόσφατη ημερομηνία που το έγγραφο του αρχείου ανανεώθηκε με κάποια νέα πληροφορία.

- **access_requests**
  - **request_id**: μοναδική αλληλουχία αναγνώρισης (αριθμοί και γράμματα) που ανατίθεται για κάθε αίτημα πρόσβασης από το σύστημα κατά την υποβολή του αιτήματος.
  - **from**: αναγνωριστική αλληλουχία (id) του χρήστη που αιτείται πρόσβαση.
  - **to**: αναγνωριστική αλληλουχία (id) του χρήστη-ιδιοκτήτη του αρχείου για το οποίο πραγματοποιήθηκε η αίτηση πρόσβασης.
  - **for**: αναγνωριστική αλληλουχία (id) του του αρχείου για το οποίο πραγματοποιήθηκε η αίτηση πρόσβασης.

5. Συμπεράσματα και Μελλοντικές Προεκτάσεις

Στο τελευταίο αυτό κεφάλαιο πραγματοποιείται η περιγραφή των αποτελεσμάτων και συμπερασμάτων που εξήχθησαν κατά την έρευνα του θέματος και την δημιουργία της εφαρμογής κατανεμημένης διαχείρισης και διασφάλισης προσωπικών δεδομένων. Πιο συγκεκριμένα, αναλύεται η γενική εικόνα της εφαρμογής, τα σχετικά πλεονεκτήματα και μειονεκτήματα που έγιναν εμφανή μετά τη χρήση των επιλεγμένων τεχνολογιών και τεχνικών και τέλος κάποιες προτάσεις για μελλοντικές προεκτάσεις της εφαρμογής, προσθέτοντας κάποιες χρήσιμες λειτουργίες.

5.1 Συμπεράσματα

Όπως έχει αναφερθεί ήδη, ο βασικός στόχος της εφαρμογής είναι η δημιουργία ενός συστήματος που δίνει την δυνατότητα στους χρήστες του να αποθηκεύουν τα δεδομένα τους και να μοιράζονται πληροφορίες με άλλους χρήστες της εφαρμογής. Ιδιαίτερη βαρύτητα έχει το ζήτημα της ιδιοκτησίας των δεδομένων και ο απώτερος σκοπός είναι η ομαλή και αποδοτική λειτουργία του συστήματος ώστε να προσφέρει τις υπηρεσίες του γρήγορα και έμπιστα.

Πρόκειται για έναν ενδιαφέρον κλάδο με σημαντικές προοπτικές ανάπτυξης, κάτι το οποίο γίνεται εμφανές από τον μεγάλο αριθμό εφαρμογών με ανάλογους στόχους που δημιουργούνται τα τελευταία χρονιά αλλά και το ενδιαφέρον των ήδη υπαρχόντων εφαρμογών για την ανακατεύθυνση τους προς την κατανεμημένη διαχείριση δεδομένων και βαρύτητα στο ζήτημα της ιδιοκτησίας δεδομένων. Ακόμα, παρατηρείται η αλλαγή στάσης σχετικά από τις ίδιες τις νομοθετικές αρχές και τους μεγάλους οργανισμούς που διαχειρίζονται μεγάλους όγκους δεδομένων των χρηστών τους. Επομένως, πριν την υλοποίηση της εφαρμογής προηγήθηκε η παρατήρηση και μελέτη των ήδη διαθέσιμων εφαρμογών στην αγορά, ώστε να αποφασιστεί ποιες είναι η προτεινόμενες τεχνολογίες και λειτουργίες για την επίτευξη των στόχων του συστήματος.

Παρατηρήθηκε πως η υλοποίηση της τεχνικής της κατανεμημένης διαχείρισης δεδομένων έχει τη δυνατότητα να προσφέρει άμεση εξυπηρέτηση των χρηστών, κάτι το οποίο είναι ιδιαίτερα σημαντικό λαμβάνοντας υπ’ όψιν τον μεγάλο όγκο δεδομένων που καταγράφονται καθημερινά και οι χρήστες επιθυμούν να διαχειρίζονται. Αποφασίστηκε

πως κατά το πρώτο στάδιο της ανάπτυξης της εφαρμογής το σύστημα θα διαχειρίζεται αριθμητικά δεδομένα για λόγους πολυπλοκότητας. Η παραπάνω επιλογή κάνει το σύστημα ιδιαίτερα χρήσιμο για δεδομένα προερχόμενα από στατιστικές έρευνες ή γενικότερα ερευνητικούς σκοπούς όπου η καταγραφή αριθμητικών δεδομένων συνηθίζεται. Προφανώς, η επιλογή αυτή δεν αποτρέπει την χρήση της εφαρμογής για οποιαδήποτε άλλο σκοπό εφόσον καλύπτονται η ανάγκες του χρήστη. Η δυνατότητα αποθήκευσης και διαχείρισης και άλλων ειδών δεδομένων (κείμενο, εικόνες, πολυμέσα, κ.α.) αποτελεί μια πολύ χρήσιμη μελλοντική προέκταση της εφαρμογής όπου αναφέρεται και παρακάτω.

Επιπρόσθετα, η μελέτη σχετικά με την ιδιοκτησία των δεδομένων σε συνδυασμό με την επιλογή της κατανεμημένης διαχείρισης δεδομένων οδήγησε στο συμπέρασμα πως το ζήτημα της συνέπειας των δεδομένων μεταξύ των κόμβων του συστήματος είναι μια ιδιαίτερα σημαντική πλευρά του συστήματος και το συγκεκριμένο χαρακτηριστικό του συστήματος θα πρέπει να αντιμετωπιστεί με την σοβαρότητα και την βαρύτητα που του αρμόζει. Ο κάθε κόμβος ωφελεί να είναι ενημερωμένος για πιθανές αλλαγές που συνέβησαν στους γειτονικούς του κόμβους ώστε να μην υπάρχει η πιθανότητα μη-εξουσιοδοτημένης χρήσης δεδομένων από χρήστες που δεν έχουν τα σχετικά δικαιώματα.

Η χρήση της εφαρμογής επισήμανε την ανάγκη για τη γρήγορη και ευχάριστη εμπειρία του χρήστη. Η παρουσίαση των δεδομένων με κατανοητό τρόπο είναι ένα σημαντικό βήμα για την καλύτερη και πιο ουσιώδη χρήση της εφαρμογής. Η δομή της εφαρμογής και τα τεχνικά χαρακτηριστικά της αναμφισβήτητα παίζουν σημαντικό ρόλο στην ανάπτυξη της εφαρμογής, αλλά καθώς πρόκειται για ένα σύστημα που έχει σχεδιαστεί με σκοπούς άμεσης αλληλεπίδρασης με τους χρήστες του, η διευκόλυνση του χρήστη με οποιοδήποτε τρόπο κρίνεται απαραίτητη. Η προσθήκη φίλτρων και περισσότερων διαθέσιμων ενεργειών για την αποδοτικότερη διαχείριση των δεδομένων από την πλευρά των χρηστών αποτελεί μια εξίσου σημαντική μελλοντική προέκταση που θα αναλυθεί παρακάτω σε συνδυασμό με την ανάπτυξη της λειτουργίας των αναφορών (reports) δεδομένων. Σε τελική ανάλυση, πρόκειται για μια εφαρμογή που ως στόχο έχει την διαχείριση μεγάλων όγκων δεδομένων, επομένως η οποιαδήποτε λειτουργία εξατομίκευσης πληροφοριών θα συντελέσει ως βελτίωση του συστήματος και της εμπειρίας του χρήστη με την εφαρμογή.

5.2 Πλεονεκτήματα και Μειονεκτήματα κατά την ανάπτυξη του συστήματος

Κατά την διαδικασία υλοποίησης και χρήσης της εφαρμογής παρατηρήθηκαν κάποια σημαντικά πλεονεκτήματα σε σχέση με άλλες παρόμοιες εφαρμογές ανάλογης κλίμακας. Μετά την ολοκλήρωση αυτά που εντοπίστηκαν είναι τα εξής:

- Εύχρηστη, φιλική προς το χρήστη και λειτουργική εφαρμογή. Ο χρήστης μπορεί να χρησιμοποιήσει την εφαρμογή με άνεση χωρίς να απαιτούνται γνώσεις κάποιας συγκεκριμένης τεχνολογίας ή γνώσεις πληροφορικής.
- Ταχύτατη εξυπηρέτηση των χρηστών χάρις της κατανομή του φόρτου εργασίας σε πολλαπλούς κόμβους. Ακόμα, με την τεχνική της διατήρησης της προσωρινής μνήμης σε κάθε κόμβο επιτυγχάνονται ακόμα πιο γρήγορη εξυπηρέτηση καθώς δεν απαιτείται η καθολική αναζήτηση δεδομένων για την εξυπηρέτηση του κάθε αιτήματος.
- Ευελιξία όσον αφορά το ζήτημα της κοινής χρήσης και ιδιοκτησίας δεδομένων. Ο χρήστης έχει την επιλογή να επιτρέψει την κοινή χρήση κάποιου δεδομένου με κάποιον άλλο χρήστη αλλά παράλληλα μπορεί να αφαιρέσει τα συγκεκριμένα δικαιώματα ανά πάσα στιγμή.
- Καθώς πρόκειται για ένα κατανεμημένο σύστημα υπάρχει πάντα η δυνατότητα επεκτασιμότητας εάν θεωρηθεί αναγκαίο χωρίς ιδιαίτερες αλλαγές.
- Οι βασικές τεχνικές ασφαλείας έχουν ληφθεί υπ' όψιν και υλοποιηθεί κατά την ανάπτυξη της εφαρμογής για να διασφαλιστεί η ασφαλή χρήση της εφαρμογής και η προστασία των πληροφοριών που καταγράφονται.

Πέρα από τα πλεονεκτήματα ωστόσο, το σύστημα εμφανίζει και ορισμένα μειονεκτήματα:

- Η επικοινωνία μεταξύ κόμβων αναδείχθηκε μη-ιδανική επιλογή κατά την ανάπτυξη του συστήματος, γεγονός που θέτει κάποια όρια στον τρόπο διαχείρισης των δεδομένων και τα ποσοστά αποδοτικότητας που θα μπορούσαν να επιτευχθούν.
- Η εφαρμογή δεν υποστηρίζει την real-time αναπαράσταση των δεδομένων, επομένως ο χρήστης χρειάζεται να ανανεώνει χειροκίνητα την εφαρμογή σε ορισμένες περιπτώσεις για να έχει πρόσβαση στα ενημερωμένα δεδομένα.
- Απουσιάζουν κάποιες λειτουργίες που θα καθιστούσαν το σύστημα ως ένα πλήρες προϊόν. Πολλές από αυτές περιγράφονται στην επόμενη παράγραφο καθώς αποτελούν μελλοντικές προεκτάσεις και βελτιώσεις της τωρινής εφαρμογής.

5.3 Μελλοντικές προεκτάσεις και προτάσεις εξέλιξης

Κατά κοινή ομολογία, η κατανεμημένη εφαρμογή που αναπτύχθηκε κατά τα πλαίσια της παρούσας πτυχιακής εργασίας, διαθέτει μεν μια συμπαγή και λειτουργική μορφή, αλλά μπορεί να εμπλουτιστεί με νέα στοιχεία και ορισμένες προεκτάσεις των τρεχόντων λειτουργιών.

5.3.1 Προτάσεις σχετικά με τα δεδομένα

Όπως αναφέρθηκε παραπάνω, στο παρόν στάδιο ανάπτυξης η εφαρμογή υποστηρίζει μόνο αριθμητικά δεδομένα ως το κύριο τύπο δεδομένων. Το παραπάνω γεγονός περιορίζει το εύρος των πιθανών χρήσεων της εφαρμογής, καθώς σε πολλές περιπτώσεις η διαχείριση άλλων ειδών πληροφοριών είναι επιθυμητή. Πρόκειται για μια σχετικά εύκολη αναβάθμιση, η οποία αποφεύχθηκε λόγω περιορισμένου διαθέσιμου χρόνου ανάπτυξης και θεωρήθηκε καλύτερη επιλογή όσον αφορά την πολυπλοκότητα. Ωστόσο, αποτελεί μια ιδιαίτερα χρήσιμη μελλοντική αναβάθμιση που θα καταστήσει την εφαρμογή ένα πλήρες σύστημα.

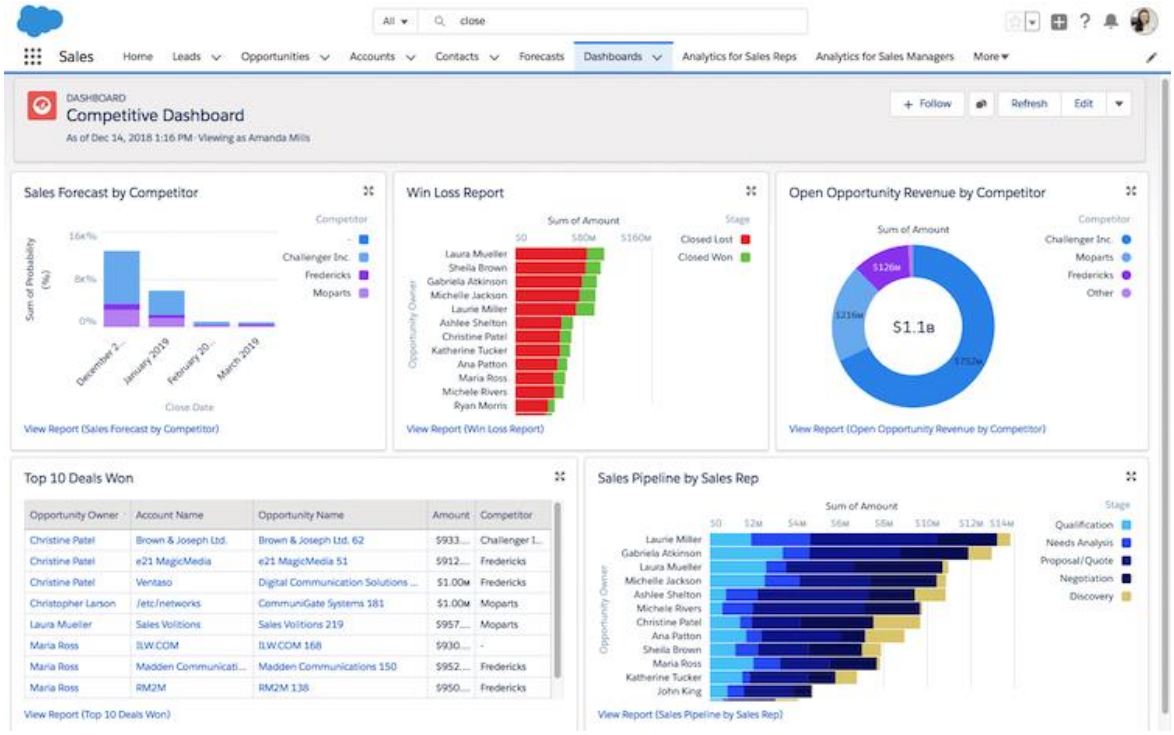
Επιπρόσθετα, όσον αφορά την παρουσίαση των δεδομένων, η υλοποίηση τεχνικών για real-time αναπαράσταση των πληροφοριών θα μετέτρεπε ριζικά την εμπειρία των χρηστών θετικά προς μια πιο σύγχρονη εικόνα του συστήματος.

Με ανάλογο τρόπο, η εξατομίκευση των παρουσιαζόμενων πληροφοριών μέσω των ανάλογων φίλτρων δεδομένων είναι μια ενδιαφέρον αναβάθμιση του συστήματος. Έχοντας υπ’ όψιν πως το σύστημα είναι σχεδιασμένο με σκοπό την διαχείριση μεγάλων όγκων δεδομένων, η χρήση φίλτρων για την εμφάνιση συγκεκριμένων δεδομένων μπορεί να αποτελέσει σπουδαίο εργαλείο για την παρουσίαση των δεδομένων στην πλευρά του χρήστη.

5.3.2 Αναφορές (Reports)

Μετά από σχετική έρευνα και μελέτη των ήδη υπαρχόντων και πετυχημένων παρόμοιων εφαρμογών, παρατηρήθηκε η σημασία της ανάλυσης των πληροφοριών. Η κατανόηση των δεδομένων και οι μεταγενέστερες γνώσεις που είναι δυνατόν να αποκτηθούν έχουν σημαντική αξία για τους χρήστες και όχι μόνο. Συστήματα με μεγάλο όγκο δεδομένων

προσφέρουν στατιστικά αποτελέσματα και χρήσιμες πληροφορίες σχετικά με τις πληροφορίες που καταγράφουν. Ανάλογα με το είδος και την φύση των πληροφοριών που επιλέγεται να διαχειριστούν από το σύστημα, υπάρχουν πολλαπλοί τρόποι ανάλυσης αυτών και εξαγωγή πολύτιμων γνώσεων σε μορφή αναφορών. Ακόμα, σε μετέπειτα στάδιο θα μπορούσαν να προστεθούν και χρήσιμα πολυμέσα, όπως γραφικές αναπαραστάσεις, διαγράμματα, κτλ., για την εικονική παρουσίαση των αναφορών. Εξαιρετικά παραδείγματα αποτελούν υλοποιήσεις όπως αυτή της Εικόνας 41, της γνωστής εταιρείας Salesforce που προσφέρει την υπηρεσία dashboard δίνοντας πολύτιμες πληροφορίες σχετικά με τα δεδομένα των πελατών της.



Εικόνα 4120: Γραφική αναπαράσταση δεδομένων του προϊόντος Salesforce, 2021, picture from https://www.salesforce.com/content/blogs/us/en/2019/01/sales-management-dashboards.html

Πρόκειται για μια πιθανή προέκταση της εφαρμογής που δεν κατευθύνεται απαραίτητα προς το θέμα της παρούσας εργασίας, ωστόσο θα αποτελούσε μια χρήσιμη αναβάθμιση σε μεταγενέστερο στάδιο ανάπτυξης.

5.3.2 Προτάσεις σχετικά με την εγγραφή και σύνδεση στο σύστημα

Αναμφισβήτητα η παρούσα διαδικασία εγγραφής νέων χρηστών στο σύστημα και η σύνδεση τους στο σύστημα είναι λειτουργική και φιλική προς τους χρήστες, ωστόσο οι πλειονότητα των σύγχρονων εφαρμογών προσφέρουν την δυνατότητα εγγραφής και

σύνδεσης μέσω της χρήσης εξωτερικών εφαρμογών, όπως της Google, Facebook, κτλ. Η παραπάνω επιλογή διευκολύνει σημαντικά την διαδικασία για τους χρήστες, καθώς μπορούν να έχουν πολλαπλούς λογαριασμούς συνδεδεμένους μεταξύ τους. Ακόμα, η εφαρμογή επωφελείται μέσω επιπλέον πληροφοριών που λαμβάνουν από τις εξωτερικές εφαρμογές και έτσι έχουν την δυνατότητα να σχηματίσουν μια πιο ολοκληρωμένη εικόνα για τους χρήστες της.

Τέλος, στα πλαίσια της σύνδεσης του χρήστη στο σύστημα, μια χρήσιμη αναβάθμιση θα ήταν η διπλή ταυτοποίηση του χρήστη με την χρήση μιας εξωτερικής authenticator εφαρμογής. Πρόκειται για μια προέκταση που θα διασφαλίσει την αυθεντικοποίηση του χρήστη και θα προσθέσει ένα επιπλέον επίπεδο ασφαλείας στο σύστημα, ζήτημα το οποίο αποτελεί ένα σημαντικό θέμα που οι οργανισμοί τείνουν να λαμβάνουν υπ’ όψιν τα τελευταία χρόνια.

Βιβλιογραφία

Ακολουθούν οι βιβλιογραφικές αναφορές (πηγές) της Εργασίας.

(n.d.).

Srinath, K. R. (2017, Dec). Python -The Fastest Growing Programming Language. *International Research Journal of Engineering and Technology (IRJET), 4*(12), 355. Retrieved from https://d1wqtxts1xzle7.cloudfront.net/61651202/IRJET-V4I126620200101-109100-1nbuifw-with-cover-page-v2.pdf?Expires=1628474089&Signature=bfgzAvVuMm6PEflHE~SYYKF01xiD15iMskXQ7d308j8o4mLYg6ga3s9i8VRdadKj-h37uKDvGUQ0~302rp9Hhl9cRflGPV0SuL~eiGsR0bNPLP0o42~5VV7

Anthony, R. J. (2016). *Systems Programming Designing and Developing Distributed Applications.* USA: Elsevier Inc.

Christensson , P. (2016, June 17). *Client-Server Model Definition*. Retrieved from TechTerms: https://techterms.com/definition/client-server_model

Comparing Service-Oriented and Distributed Object Architectures. (2005). In S. Baker, & S. Dobson, *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE* (Vol. 3760, pp. 631-645). Berlin: Springer. doi:https://doi.org/10.1007/11575771_40

Coulouris, G., Dollimore, J., Kindberg, T., & Blair, G. (2001). *DISTRIBUTED SYSTEMS Concepts and Design* (4th Edition ed.). (I. Pearson Education, Ed.) UK: Addison-Wesley. Retrieved from https://ce.guilan.ac.ir/images/other/soft/distribdystems.pdf

Dauzon, S., Bendoraitis, A., & Ravindran, A. (2016). What is django? In *Django: Web Development with Python* (pp. 5-8). Packt Publishing Ltd.

Distributed Objects. (2000). In H. Balen, M. Elenko, J. Jones, & G. Palumbo, *Distributed Object Architectures with CORBA.* Cambridge University Press.

Faisandier, A., Roedler, G., & Adcock, R. (2021). *System Architecture*. Retrieved from sebokwiki: https://www.sebokwiki.org/wiki/System_Architecture

Get started. (2021, August 06). Retrieved from jetbrains.com: https://www.jetbrains.com/help/pycharm/quick-start-guide.html

Ghimire, D. (2020). Django. *Comparative study on Python web frameworks: Flask and Django*, 9.

Ghosh, S. (2014). *Distributed Systems An Algorithmic Approach* (Second ed.). Iowa City, Iowa, USA: Taylor & Francis Group. Retrieved from https://books.google.co.uk/books?hl=en&lr=&id=60fSBQAAQBAJ&oi=fnd&pg=PP1&dq=Distributed+Systems+An+Algorithmic+Approach+Second+Edition&ots=W7jJpKSnrh&sig=XiynCrQDUadONKfslbHMCTvg070#v=onepage&q=Distributed%20Systems%20An%20Algorithmic%20Approach%20Second%

Meier, R., & Gross, T. R. (2019, October 20). Reflections on the Compatibility, Performance, and Scalability of Parallel Python. (pp. 1-2). New York, NY, USA: Association for Computing Machinery. doi:10.1145/3359619.3359747

Multithreading vs Multiprocessing: What's the difference? (n.d.). Retrieved from guru99: https://www.guru99.com/difference-between-multiprocessing-and-multithreading.html#2

Neuman, C. B. (1994). Scale in Distributed Systems. 28. Retrieved from http://clifford.neuman.name/papers/pdf/94--_scale-dist-sys-neuman-readings-dcs.pdf

Oluwatosin, H. S. (2014). Client-Server Model. *IOSR Journal of Computer Engineering (IOSR-JCE), 16*(1), 67-71. Retrieved from https://www.researchgate.net/profile/Shakirat-Sulyman/publication/271295146_Client-Server_Model/links/5864e11308ae8fce490c1b01/Client-Server-Model.pdf

Özsu, T. M., & Valduriez, P. (2011). *Principles of Distributed Database Systems* (3 ed.). Springer, Cham. doi:https://doi.org/10.1007/978-3-030-26253-2

Panjwani, S. (2018, June 5). *Multiprocessing Vs. Threading In Python: What You Need To Know*. Retrieved from timber.io: https://timber.io/blog/tag/multiprocessing/

Pubudumal, D. (2020, January 5). *CAP Theorem in Distributed Systems*. Retrieved from Medium: https://medium.com/swlh/cap-theorem-in-distributed-systems-edd967e7bdf4

Python Documentation by Version. (2021). Retrieved from python.org: https://www.python.org/doc/versions/#:~:text=Python%202.7%2C%20documentation%20released%20on,released%20on%2029%20October%202013.

Rubio, D. (2017). Introduction to the Django Framework. In *Beginning Django.* Ensenada, Mexico: APRESS. doi:https://doi.org/10.1007/978-1-4842-2787-9

Shah, R. (2021, Jun 29). *Fragmentation in Distributed Database Management System (DDBMS)*. Retrieved from benchpartner: https://benchpartner.com/fragmentation-in-distributed-database-management-system-ddbms

Sommerville, I. (2006). Αρχιτεκτονικές κατανεμημένων συστημάτων. Στο I. Sommerville, *Βασικές αρχές Τεχνολογίας Λογισμικού* (8 εκδ.). Pearson Education Limited.

Steen, M. v., & Tanenbaum , A. S. (2016, October). A brief introduction to distributed systems. *A brief introduction to distributed systems*, 967–1009. doi:https://doi.org/10.1007/s00607-016-0508-7

Steen, V. M., & Tanenbaum, A. (2002). *Distributed systems principles and paradigms.* Network 2.

Tanenbaum, A. S., & Steen, M. V. (2006). *Distributed Systems Principles and Paradigms* (Second ed.). (Marcia J. Horton, Ed.) Amsterdam, Netherlands: Pearson Education Inc.

Thakur, D. (n.d.). *What is the difference between Homogeneous and Heterogeneous Distributed DBMSs*. Retrieved from Computer-Notes, Inc: https://ecomputernotes.com/database-system/adv-database/homogeneous-and-heterogeneous-distributed-dbmss

Varun, K. G. (2020, April 13). *DIY: Multithreading vs Multiprocessing in Python*. Retrieved from gitconnected.com: https://levelup.gitconnected.com/diy-multithreading-vs-multiprocessing-in-python-fb93698ca7f3

What is PyCharm? (2021, Junε 15). Retrieved from intellipaat.com: https://intellipaat.com/blog/what-is-pycharm/

What is Python? Executive Summary. (n.d.). Retrieved from python.org: https://www.python.org/doc/essays/blurb/

Whittaker, M. (n.d.). *An Illustrated Proof of the CAP Theorem.* Retrieved from https://mwhittaker.github.io/blog/an_illustrated_proof_of_the_cap_theorem/.

Wiese, L. (2014). Clustering-based fragmentation and data replication for flexible query answering in distributed databases. *Cloud Computing*, *3*(18). doi:https://doi.org/10.1186/s13677-014-0018-0

Zhang, H. (2013, Jul 25). Architecture of Network and Client-Server model. *Networking and Internet Architecture (cs.NI)*. doi:arXiv:1307.6665 [cs.NI]

Ziadé, T. (2008). Multithreading. In T. Ziadé, & S. Kulkarni (Ed.), *Expert Python Programming* (pp. 306-309). Birmingham: Packt Publishing. Retrieved from https://www.e-reading-lib.com/bookreader.php/138816/Expert_Python_programming.pdf

Υπεύθυνη Δήλωση Συγγραφέα:

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν. 1599/1986 και τα άρθρα 2,4,6 παρ. 3 του Ν. 1256/1982, η παρούσα εργασία αποτελεί αποκλειστικά προϊόν προσωπικής εργασίας και δεν προσβάλλει κάθε μορφής πνευματικά δικαιώματα τρίτων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον.